საქართველოს ტექნიკური უნივერსიტეტი

დავით ჭოხონელიძე

მანქანური სწავლების კომბინირებული ალგორითმების დამუშავება

წარმოდგენილია დოქტორის აკადემიური ხარისხის მოსაპოვებლად

სადოქტორო პროგრამა “ინფორმატიკა”, შიფრი 0401

საქართველოს ტექნიკური უნივერსიტეტი

თბილისი, 0175, საქართველო

ივლისი, 2016 წელი

თბილისი
2016 წელი

სამუშაო შესრულებულია საქართველოს ტექნიკური უნივერსიტეტში
ინფორმატიკისა და მართვის სისტემების ფაკულტეტი
მართვის ავტომატიზებული სისტემების დეპარტამენტი

ხელმძღვანელი: პროფ. ზურაბ ბოსიკაშვილი

რეცენზენტები: ----------

დაცვა შედგება ----2016------ წლის ”--8----” ------07----------, --------- საათზე
საქართველოს ტექნიკური უნივერსიტეტის ----------
---------- ფაკულტეტის სადისერტაციო საბჭოს კოლეგიის
სხდომაზე, კორპუსი ----------, აუდიტორია ----------
მისამართი: 0175, თბილისი, კოსტავას 77.

დისერტაციის გაცნობა შეიძლება სტუ-ს ბიბლიოთეკაში,
ხოლო ავტორეფერატისა - ფაკულტეტის ვებგვერდზე

სადისერტაციო საბჭოს მდივანი პროფ. თინათინ კაიშაური

საქართველოს ტექნიკური უნივერსიტეტი

ინფორმატიკისა და მართვის სისტემების ფაკულტეტი

ჩვენ, ქვემორე ხელისმომწერნი ვადასტურებთ, რომ გავეცანით დავით ჭოხონელიძის მიერ შესრულებულ სადოქტორო ნაშრომს დასახელებით: „მანქანური სწავლების კომბინირებული ალგორითმების დამუშავება“ და ვაძლევთ რეკომენდაციას საქარველოს ტექნიკური უნივერსიტეტის ინფორმატიკისა და მართვის სისტემების ფაკულტეტის სადისერტაციო საბჭოში მის განხილვას დოქტორის აკადემიური ხარისხის მოსაპოვებლად.

თარიღი

ხელმძღვანელი: პროფ. ზურაბ ბოსიკაშვილი

რეცენზენტი:

რეცენზენტი:

საქართველოს ტექნიკური უნივერსიტეტი

2016

ავტორი: დავით ჭოხონელიძე

დასახელება: „მანქანური სწავლების კომბინირებული ალგორითმების დამუშავება“;

ფაკულტეტი : ინფორმატიკისა და მართვის სისტემების

ხარისხი: დოქტორი

სხდომა ჩატარდა:

ავტორის ხელმოწერა

ნაშრომს ვუძღვნი ჩემს უსაყვარლეს მეუღლესა და ოჯახს ჩემთვის გაწეული ფსიქოლოგიური და მორალური დახმარებისთვის.

რეზიუმე

მანქანური სწავლების დანიშნულებაა რაიმე სისტემის (ობიექტის) შემავალი და გამომავალი სიდედეების, მდგომარეობების დაკვირვებათა ნაკრებიდან შედგეს ამ სისტემის ან ობიექტის მოდელი, რომლის საშუალებითაც შესაძლებელი იქნება სხვდასხვა ამოცანების (პროგნოზირება, გადაწყვეტილების მიღება და სხვა) გადაწყვეტა. ინტელექტუალური სისტემები ეფუძვნება მანქანურ სწავლებას (სხვა მნიშვნელოვან ასპექტებთან ერთად) აქედან გამომდინარე უდიდესი მნიშვნელობა აქვს შესაბამისი მოდელის აგებას. არსებობს უამრავი სხვადასხვა ტიპის ინტელექტუალური სისტემა: მათემატიკური, ბიოლოგიური და ა.შ. ასეთი ტიპის სისტემები შეიძლება გათვლილი იყოს მხოლოდ კონკრეტული ტიპის ინფორმაციაზე. არსებობს უამრავი სხვადასხვა ტიპის ალგორითმი, რომლებიც გამოიყენება ასეთ სისტემებში, მაგ: კლასიფიკაციის, კლასტერიზაციის ტიპის ალგორითმები. როგორც წესი მათი ცალ ცალკე გამოყენება ვერ იძლევა ეფექტურ შედეგს, აქედან გამომდინარე საჭირო ხდება ასეთი ტიპის ალგორითმების დაჯგუფება, ერთმანეთში კომბინაცია (კომბინაცია არ გულისხმობს მხოლოდ ალგორითმების ერთმანეთში შეერთებას და ახალი ალგორითმის მიღებას, არამედ შეიძლება გულისხმობდეს მკაცრად განსაზღვრული მიმდევრობით რამოდენიმე ტიპის ალგორითმის გამოყენებას ერთ მთლიან სისტემაში). ნებისმიერ შემთხვევაში აუცილებელია შემდეგი მნიშვნელოვანი საკითხების განსაზღვრა : შემავალი და გამომავალი ინფორმაცია, სისტემის მუშაობის ძირითადი პროცესები და პრინციპები. ნებისმიერი გამოყენებული ალგორითმისათვის აუცილებელია ინფორმაციის ადექვატურად მიწოდება. ინტელექტუალური სისტემა შეიძლება მრავალნაირი იყოს, აქედან გამომდინარე სრულიად შესაძლებელია შემავალი ინფორმაცია იყოს ნებისმიერი სახით წარმოდგენილი. ინფორმაციის ტიპები მრავალნაირია, შესაბამისად თავდაპირველად

მოწოდებული შემავალი ინფორმაცია შეიძლება იყოს სხვადასხვა ტიპის: ტექსტური , რიცხვითი და ა.შ. თუმცა ინტელექტუალური ტიპის სისტემა ძირითადად ემყარება ორი ტიპის ინფორმაციას, რომელთაგან პირველია რაოდენობრივი ხოლო მეორე ხარისხობრივი (ისინი თავიანთ თავში აერთიანებენ სხვადასხვა ქვე ტიპის ინფორმაციას). სისტემა შეიძლება იყენებდეს ისეთი ტიპის ალგორითმებს, რომლებიც რეალიზებულია მხოლოდ ბინარული ფორმატისათვის, აქედან გამომდინარე აუცილებელია მოხდეს თავდაპირველად შემავალი ინფორმაციის ნორმალიზება და ალგორითმისათვის შესაბამისად მიწოდება. ასევე მნიშვნელოვანია გამომავალი ინფორმაციის დენორმალიზაციაც (დენორმალიზაცია გულისმობს მაქსიმალური სიზუსტით შესაბამისი თავდაპირველი მნიშვნელობების აღდგენას). მისი საჭიროება ნათლად ჩანს ისეთი ტიპის ინტელექტუალურ სისტემებში, რომლებიც ადამიანის ჩაურევლად ღებულობენ და აწვდიან ადამიანისათვის გასაგებ უკვე მზა გადაწყვეტილებას (თუ ასეთი გადაწყვეტილება რამოდენიმეა მაშინ ადამიანი ახდენს ამორჩევას). რა თქმა უნდა თუკი ასეთი სისტემები არ მოახდენენ გამომავალი ინფორმაციის დენორმალიზაციას (ადამიანისათვის გასაგებ ფორმატში წარმოდგენას), ადამიანი მეტწილად ვერ მიხვდება რა გადაწყვეტილება მიიღო სისტემამ, ამდენად არანაკლებ მნიშვნელოვანია გამომავალი ინფორმაციის დენორმალიზაციაც. ინტელექტუალურ სისტემებში, როდესაც გვაქვს ალგორითმების კომბინაცია, თავს იჩენს შემდეგი მნიშვნელოვანი ფაქტორი, რომელიც უშუალოდ ნორმალიზაცია/დენორმალიზაციასთან არის დაკავშირებული: ერთი ალგორითმის გამომავალი ინფორმაცია შეიძლება გამოყენებულ იქნას როგორც შემავალი ინფორმაცია სხვა ალგორითმებისათვის. შესაბამისად როდესაც სისტემამ ეს იცის, აუცილებელია გაითვალისწინოს ამ გამომავალი ინფორმაციის შესაბამისად ნორმალიზაცია (რომელიც გამოყენებულ იქნება სხვა ალგორითმების მიერ). როდესაც ერთი ალგორითმის გამომავალი ინფორმაცია გამოიყენება ერთზე მეტი სხვადასხვა ტიპის

ალგორითმისათვის, აუცილებელია თითოეული ტიპის ალგორითმისათვის შესაბამისად მოვახდინოთ ინფორმაციის ნორმალიზაცია. ინტელექტუალური სისტემის ერთ-ერთი მნიშვნელოვანი ფაქტორია ინფორმაციის სიღრმისეული ანალიზის საფუძველზე მათი შენახვა ე.წ Data Mining [7]. მონაცემთა ასეთი განაწილება საშუალებას აძლევს ინტელექტუალურ სისტემას გააკეთოს გარკვეული ტიპის დასკვნები საბოლოო შედეგიდან გამომდინარე. დასკვნის გაკეთებას აქვს თავისი სირთულეც, თუმცა ის მაინც დამოკიდებულია ისეთ ძირითად ფაქტორებზე, როგორციაა: ამოცანის შინაარსი და სპეციფიკა, გარკვეული გარემო პირობები რაც აუცილებელია დასკვნის გასაკეთებლად (ან მიახლოებითი სიზუსტით დასკვნის გასაკეთებლად) და ა.შ. ინტელექტუალური სისტემის სპეციფიკიდან გამომდინარე, საბოლოო დასკვნის გასაკეთებლად შეიძლება ვერ მოხერხდეს მხოლოდ საბოლოო შედეგის საფუძველზე და ასევე საჭირო გახდეს მთლიანი პროცესის დეტალური ანალიზი. ამ ანალიზში შედის: ალგორითმების მიმდევრობაზე დაკვირვება, თითოეული ალგორითმის ყოველ ბიჯზე მიღებულ შედეგებზე დაკვირვება, ალგორითმების გამოძახებების სიხშირე (ეს მეტწილად მაინც დამოკიდებულია შემავალ ინფორმაციასა და ამოცანის სპეციფიკაზე) . დამატებით შეიძლება არსებობდეს უამრავი სუბიექტური მოვლენა, რომელზე განსაზღვრა და დაკვირვება შეიძლება ვერ მოახდინოს სისტემამ, აქედან გამომდინარე ადამიანს (რომელიც აღნიშნულ სისტემას მოიხმარს) უნდა შეეძლოს გარკვეული მცირე ანალიზის საფუძველზე მიახლოებით მაინც მიხვდეს სისტემა რამდენად ადექვატურ და სწორ გადაწყვეტილებას სთავაზობს. მიმდინარე ნაშრომის მიზანია შექმნას ინტელექტუალურ სისტემაში გამოყენებადი მოდელი, რომელიც ეფუძვნება მანქანურ სწავლებას და რომელიც აგებულია შემდეგ ძირითად ჯაჭვზე : შემავალი ინფორმაცია - > ინფორმაციის ნორმალიზაცია -> ალგორითმები (ინფორმაციის ნორმალიზაცია/დენორმალიზაცია ალგორითმების

ერთმანეთთან სწორად დასაკავშირებლად) -> გამომავალი ინფორმაცია -> გამომავალი ინფორმაციის დენორმალიზაცია -> გარკვეული დასკვნა.

Abstract

The main purpose of machine learning is to create a model from input and output quantities, state of training set which will be used to solve some kind of problems (prediction, decision making and etc.). Intelligence systems are based on machine learning (with other important aspects). Therefore it's important to create an adequate model. There exists many kind of intelligence systems: Mathematical, Biological etc. Such kind of systems might be based on some kind of information. There exists a lot of different algorithms which are used for such systems, for example: Classification, Clustering. As usual if they are used separately, it does not affect high performance. It appears from this we have to combine/merge such kind of algorithms (Combining does not mean producing new algorithm, but it defines a sequence of any kind algorithms which will be used in whole system). In any case it's required to define these important questions: Input and output information, the main processes and principles of whole system. It's important to define input information adequately for every used algorithm. There exists many kind of intelligence system, therefore input information might be defined in any format. There exists many kind of information, therefore it may be: String based, Number based and etc. As usual intelligence systems are based on two kinds of information: Quantitative and Qualitative (In them any other kind of information are united). System may be based on such kind of algorithms which are based on binary format, therefore it is required to normalize input data before passing it to algorithm. It's also important to denormalize output information (Denormalization means to restore output information in original value with adequate accuracy). The usage of denormalization is clear in such kind of intelligence systems which makes decision without human (If there exists multiple decisions, system suggests human to select any of them). Of course if such systems won't denormalize output information (In human readable format), human won't be able to understand output decision, so denormalization is not less important than normalization. In

intelligence systems, when we are having combinations of algorithms, there exists one important factor which is related to normalization/denormalization: Output information of one algorithm may be used as input data for another one. Therefore when system knows this, it is required to normalize output data (In order to use it for another algorithm). It is important to normalize data when we use algorithm's output data as an input data for another one. Data Mining [7] is one of the most important part of intelligence system. Such definition of data makes intelligence system to draw a conclusion with using final results. It is not easy to make final decision, it is depend of many factors, for example : problem description and specification, environment which is important in order to make decision (or approximately accuracy conclusion) and etc. Whole process analyze may be required in order to make final decision with using specification of intelligence system. This analyze contains: Observation of algorithm sequence, Observation of each algorithm's each step and their results, Observation of quantity of algorithm invocation (This mostly depends on input data and specification of algorithm). Additionally there may exist a lot of events on which system could not observe, therefore human (Who uses this system) must be able to analyze some events and guess what decision had system made, how adequate is this solution. The main purpose of this PHD is to create model which will be used in intelligence systems, which will be based on machine learning and will be represented as following chain: Input information/data -> algorithms (Information normalization/denormalization in order to communicate algorithms to each other) -> Output Information -> Denormalization of output information -> Final decision.

შინაარსი

შესავალი 17
I. სამუშაოს მიზანი 20
II. კვლევის ძირითადი ამოცანები 22
III. კვლევის ობიექტი 23
IV. პრაქტიკული მნიშვნელობა 24
1. არსებული სისტემების ზოგადი მიმოხილვა 25
2. ერთ-ერთი ძირითადი ამოცანა 35
3. მეთოდები და ალგორითმები 52
4. ექსპერიმენტი კონკრეტულ მონაცემებზე 86
5. სისტემის პროგრამული მოდელის სახით წარმოდგენა 89
დასკვნა 107
გამოყენებული ლიტერატურა 108

ნახაზების ნუსხა

სურ 1. ინფორმაციის დასწავლა 26

სურ. 2 ინფორმაციის ამოშლა 27

სურ. 3 ინტერნეტში ინფორმაციის მოძიება 28

სურ. 4 მათემატიკური ოპერაციები 29

სურ. 5 Mycin ის არქიტექტურა 30

სურ. 6 საკონსულტაციო სისტემის გამოყოფა 31

სურ. 7 სტატიკური ბაზის გამოყოფა 32

სურ. 8 დინამიური ბაზის გამოყოფა 32

სურ. 9 - სხვადასხვა დაპრ.ენების და ტექნოლოგიების მხარდაჭერა 36

სურ. 10 სერვერების, ტექნოლოგიებისა და ბაზების მხარდაჭერა დაპრ.ენა Java ზე 37

სურ. 11 სისტემის წარმოდგენა გრაფიკულად 38

სურ. 12 ერთ-ერთი სართულის ჭრილი, სადაც ოთახების განლაგება ჩანს .. 40

სურ. 13 Wifi სიგნალის გაზომვა სპეციალური პროგრამით 41

სურ. 14 სიგნალების გრაფის ჩვენება სპეციალური პროგრამით 42

სურ. 15 ცუდი და კარგი სიგნალის სიხშირეები 43

სურ. 16 Wifi სიგნალების ხარისხის განსაზღვრა მიმართებებით 44

სურ. 17 აგენტი და მისი შესაძლო თვისებები 45

სურ. 18 ინტელექტუალური სისტემა, როგორც ერთიანი სისტემა 47

სურ. 19 ინტ.სისტემის სრულყოფილებისათვის სხვადასხვა საჭირო კომპონენტები 49

სურ. 20 სისტემა, როგორც შემავალი და გამომავალი ინფორმაციის ერთობა 50

სურ. 21 Greybox მოდელის შესაბამისი სისტემა 54

სურ. 22 აპრიორის ალგორითმი, კონკრეტული მაგალითი 56

სურ. 23 გადაწყვეტილებათა ხეები, საწყისი მატრიცა 58

სურ. 24 ავირჩიეთ კვანძი და მოვახდინეთ ორ სიმრავლედ გაყოფა საწყისი მატრიცის 59

სურ. 25 მარცხენა მატრიცადან მიღებული მნიშვნელობა 59

სურ. 26 პროცესის გაგრძელება მარცხენა მატრიცისთვის 60

სურ. 27 მატრიცის კიდევ ერთხელ გაყოფა 60

სურ. 28 პროცესის გაგრძელება დარჩენილი მატრიცისთვის 61
სურ. 29 წინა ეტაპზე მიღებული მატრიცა კვლავ გაიყო..... 61
სურ. 30 საბოლოოდ აგებული ხე..... 62
სურ. 31 გადაწყვეტილებათა ხის აგება, მთლიანი პროცესი..... 62
სურ. 32 xor ოპერაციით ხის აგება 63
სურ. 33 მატრიცის საწყისი გაყოფა..... 63
სურ. 34 მატრიცების გაყოფა (გაგრძელება) 64
სურ. 35 მატრიცის გაყოფა (გაგრძელება) 64
სურ. 36 xor ოპერაციის გამოყენებით მიღებული გადაწყვეტილებათა ხე 65
სურ. 37 კლასტერიზაციის ზოგადი სქემა..... 66
სურ. 38 K-Means ალგორითმის სიმულაცია..... 67
სურ. 39 მონაცემთა ტიპები..... 69
სურ. 40 Equilateral Encoding ის მაგალითი..... 72
სურ. 41 Equilateral Encoding ის ვიზუალური წარმოდგენა..... 73
სურ. 42 შემუშავებული ლოგიკური სქემა 74
სურ. 43 ლოგიკური სქემის გამოყენების მაგალითი..... 75
სურ. 44 საწყისი მონაცემები 76
სურ. 45 საწყისი კლასტერები..... 77
სურ. 46 კლასტერების ასაგებად მონაცემთა განაწილება 77
სურ. 47 კლასტერის გარდაქმნა..... 77
სურ. 48 მანძილების დათვლა სხვადასხვა კლასტერამდე..... 78
სურ. 49 საბოლოოდ მიღებული კლასტერი 78
სურ. 50 საბოლოოდ მიღებული გარდაქმნა ბინარულ ფორმატში..... 79
სურ. 51 კლასტერიზაციის ერთ ერთი ვიზუალიზაცია 79
სურ. 52 შემთხვევითი ინიციალიზაციის სქემა 81
სურ. 53 ცენტროიდებზე დაფუძვნენული ინიციალიზაცია 82
სურ. 54 ნორმალიზაციის ერთიანი ფორმულა 84
სურ. 55 დენორმალიზაცია (მთლიანი პროცესი)..... 84
სურ. 56 კლასიფიკაციისა და კლასტერიზაციის შედარება 85
სურ. 57 Wifi ს განლაგების კონკრეტული მაგალითი 86
სურ. 58 ექსპერიმენტის შედეგად მიღებული გადაწყვეტილებათა ხე..... 88
სურ. 59 ორ დონიანი არქიტექტურის მაგალითი..... 90

სურ. 60 სამ დონიანი არქიტექტურა 91
სურ. 61 ერთ-ერთი აპლიკაციის სერვერვის მაგალითი..... 92
სურ. 62 სერვერის კლასტერის მაგალითი 93
სურ. 63 მონაცემთა ბაზის კლასტერიზაცია 94
სურ. 64 სერვერისა და ბაზის კლასტერის ერთიანი მაგალითი 95
სურ. 65 ჩვენს მიერ შემუშავებული სისტემის ზოგადი არქიტექტურა..... 96
სურ. 66 სისტემის ზოგიერთი ტექნიკური მხარე 98

მადლიერება

დიდ მადლიერებას მოვახსენებ ჩემს ხელმძღვანელს, ბატონ ზურაბ ბოსიკაშვილს, გაწეული დახმარებისათვის

შესავალი

ხელოვნური ინტელექტი არის მეცნიერების დარგი, რომელიც ცდილობს არა მარტო შეიცნოს ინტელექტის ბუნება, არამედ შექმნას კიდეც (ზოგჯერ სწორედ ესაა უფრო მთავარი) ინტელექტუალური ქცევის მქონე ხელოვნური სისტემები, ანუ როგორც მათ უწოდებენ ხელოვნური ინტელექტუალური სისტემები.

ხელოვნური ინტელექტი მეცნიერების ერთერთი ყველაზე ახალგაზრდა დარგია. პირველი შრომები ამ მიმართულებით გაჩნდა მეორე მსოფლიო ომის შემდეგ, ხოლო ტერმინი - ხელოვნური ინტელექტი პირველად გამოიყენეს 1956 წელს. მისი წარმოშობაც და განვითარებაც მჭიდროდაა დაკავშირებული კომპიუტერული ტექნიკის წარმოშობასა და განვითარებასთან და ეს რა თქმა უნდა შემთხვევითი არაა. ყველა ის ზემოთჩამოთვლილი ქმედებები, რომლებიც განსაზღვრავენ ინტელექტს, როგორც ფენომენს, თავისი არსით წარმოადგენენ ინფორმაციულ პროცესებს, ანუ ყველა ამ ქმედების დროს ხდება ინფორმაციის მიღება, დამუშავება, შენახვა და გადაცემა. კომპიუტერული ტექნიკის წარმოშობამდე ასეთი უნარი გააჩნდათ მხოლოდ ცოცხალ ორგანიზმებს (სხვადასხვა დონეზე). კომპიუტერებმა საშუალება მოგვცეს მოგვეხდინა ამ პროცესების მექანიზაცია ტექნიკურ სისტემებში, რაც გახდა ხელოვნური ინტელექტუალური სისტემების განვითარების საფუძველი. ერთ-ერთი მნიშვნელოვანი ფაქტორი რაც აღნიშნულ დარგს ახასიათებს არის დამოუკიდებლად აზროვნება და თვითგანვითარების უნარი. აზროვნება საკმაოდ რთული პროცესია, ის მოიცავს როგორც გარკვეული ცოდნის გამოყენების უნარს ასევე გამოცდილებასაც რომელსაც ადამიანი ახმარს საკუთარ ცოდნას პრობლემის გადაჭრის დროს. რაც დრო გადის და იხვეწება თანამედროვე ტექნოლოგიები, ხელოვნური ინტელექტი უფრო და უფრო მნიშვნელოვანი ხდება. როგორც ზოგიერთი ცნობილი მეცნიერი ამბობს : „მომავალი ხელოვნური ინტელექტისაა“ . მათი ვარაუდით ძალიან მალე დადგება დრო როდესაც გარკვეული პროგრამული

უზრუნველჰყოფის წერას მთლიანად ინტელექტუალური სისტემები ჩაანაცვლებს (მათ ისიც კი შეეძლებათ, რომ პროგრამა თითონ შექმნან). მათ ექნებათ განვითარების უნარი, თავად შეისწავლიან ახალ ინფორმაციას და მის ხარჯზე უფრო „ჭკვიანები“ გახდებიან. რასაკვირველია ეს ყოველივე არც ისე იოლია და არც ადვილად მიღწევადი (თანამედროვე დროში ამ კუთხით უკვე არსებობს გარკვეული მიღწევები, მაგალითად კორპორაცია Google ის მიერ შექმნილი ავტომატურად მართვადი ავტომობილი [1]). ინტელექტუალური სისტემა თავის მხრივ მოიცავს ისეთ განუყოფელ დარგებს, როგორიცაა: მანქანური სწავლება, მონაცემთა ანალიზი და შესაბამისად შენახვა (ე.წ Data Mining ი). მანქანური სწავლება კატეგორიულ დონეზე აერთიანებს სხვადასხვა სახის ალგორითმებს, რომელთა დახმარებითაც გარკვეული შედეგების საფუძველზე უზრუნველჰყოფს სწავლებას. რაც დრო გადის ინფორმაციის ნაკადი მკვეთრად იზრდება და შესაბამისად ამ ინფორმაციის დასწავლასა და დამუშავებას ინტელექტუალური სისტემა გაცილებით ეფექტურად და ჩქარა შეძლებს ვიდრე ადამიანი. რასაკვირველია აუცილებელია ასეთი ტიპის სისტემა კარგად იყოს გამართული და გატესტილი. არის კიდევ ერთი მნიშვნელოვანი ფაქტორი, რომლის საფუძველზეც ცხადია რომ ეს დარგი კიდევ უფრო აქტუალურია, კერძოდ: რაც დრო გადის უფრო და უფრო იზრდება ისეთი ამოცანების რიცხვი, რომლის რეალიზაციაც უფრო ეფექტურია ხელოვნური ინტელექტის საშუალებით, ვიდრე რაიმე კონკრეტული პროგრამული უზრუნველჰყოფის ხარჯზე. თავისთავად ცხადია რომ ნებისმიერი ინტელექტუალური სისტემა წარმოადგენს პროგრამულ უზრუნველჰყოფას, მაგრამ არა პირიქით. როგორც უკვე აღვნიშნეთ ინტელექტუალური სისტემა თავის თავში მოიცავს მანქანურ სწავლებასა და Data Mining ს. ცხადია მათ გარეშე ინტელექტუალური სისტემა თითქმის არაფერს წარმოადგენს, აქედან გამომდინარე ამ საკითხის აქტუალობა პირდაპირ კავშირშია ისეთ დარგებთან როგორიცაა მანქანური

სწავლება და Data Mining. აღნიშნული ნაშრომი აღწერს მოდელს, რომელიც ემყარება მანქანური სწავლების ალგორითმებს.

I. სამუშაოს მიზანი

როგორც უკვე აღვნიშნეთ ინტელექტუალური სისტემა მოიცავს ისეთ ქვე დარგებს, როგორიცაა მანქანური სწავლება და Data Mining ი [7]. მანქანური სწავლება მოიცავს სხვადასხვა კატეგორიის უამრავ ალგორითმს, რომელთა გამოყენებითაც მიიღწევა ინტელექტუალური სისტემის გამართული და ეფექტური მუშაობა. ალგორითმები ერთიანდება გარკვეულ კატეგორიებად, რომელთაგან ზოგიერთი მათგანია: კლასიფიკაციის ტიპის ალგორითმები, კლასტერიზაციის ტიპის ალგორითმები , ასოციაციის ტიპის ალგორითმები და ა.შ. როგორი ტიპის ალგორითმებსაც არ უნდა იყენებდეს მანქანური სწავლება , აუცილებელია ისეთი ძირითადი ფაქტორების განსაზღვრა, როგორიცაა: შემავალი და გამომავალი ინფორმაცია, ინფორმაციის ადექვატურად ფორმირება, ალგორითმების სწორი შერჩევა, საბოლოო შედეგიდან გამომდინარე გადაწყვეტილების მიღება (არსებობს ე.წ ექსპერტული სისტემები, რომლებიც რეალურად წარმოადგენენ ინტელექტუალურ სისტემებს. ისინი საბოლოო შედეგებიდან გამომდინარე ღებულობენ გარკვეულ გადაწყვეტილებებს). როგორც წესი თითოეული კატეგორიის ალგორითმების ცალ ცალკე გამოყენება ვერ იძლევა ეფექტურ შედეგს, აქედან გამომდინარე უფრო უკეთესი იქნება თუ მოვახდენთ მათ გაერთიანებასა და კომბინაციას. ალგორითმების კომბინაციაში არ იგულისხმება მხოლოდ მათი ერთმანეთში შერწყმა, არამედ გარკვეული მიმდევრობის მიღება, რომელიც ეფექტურს გახდის მთლიან სისტემას. ამ მიზნით აუცილებელი ხდება შევარჩიოთ ერთზე მეტი ალგორითმი (თუნდაც სხვადასხვა კატეგორიიდან) და ვნახოთ თუ როგორ იმოქმედებს მათი ერთიანობა სისტემის წარმადობაზე. ალგორითმების ერთმანეთთან გადაბმის დროს უდიდეს როლს თამაშობს ინფორმაციის ნორმალიზაცია[2]. მისი საშუალებით ალგორითმები ერთმანეთს „ელაპარაკებიან“ ადექვატურ ფორმატში. სამუშაოს ძირითად მიზანს წარმოადგენს შეიქმნას ერთიანი სისტემა, რომელშიც გაერთიანებული იქნება ისეთი ალგორითმები (ალგორითმები რასაკვირველია შეიძლება მიეკუთვნებოდნენ სხვადასხვა

კატეგორიებს), რომლებიც მანქანური სწავლების დარგში შექმნიან ეფექტურ სტრუქტურას. ამ სწავლების სტრუქტურას კი დაეყრდნობა ინტელექტუალური სისტემა. საბოლოო ეტაპზე კი შეიქმნება შესაბამისი პროგრამული უზრუნველჰყოფა, რომელიც მოახდენს აღნიშნულის სიმულაციას.

II. კვლევის ძირითადი ამოცანები

ინტელექტუალური სისტემა შეიძლება იყოს უამრავი სახის: მათემატიკური, ბიოლოგიური და ა.შ. მათ ყველას უნდა ახასიათებდეს სწავლის და განვითარების უნარი. რასაკვირველია ნებისმიერი ინტელექტუალური სისტემა (როგორი სახისაც არ უნდა იყოს ის) შემოსაზღვრულია ერთი ან რამოდენიმე დარგით, მაგ: ბიოლოგიური სისტემა ეყრდნობა ბიოლოგიას, რა თქმა უნდა ის შეიძლება თავის თავში სწავლობდეს ისეთ დარგებს რომელიც თუნდაც ბიოლოგიასთან მჭიდროდაა დაკავშირებული, თუმცა მაინც არ გადის ამ ჩარჩოებიდან. საკვლევი ამოცანები უამრავია. მათთვის შეიძლება შეიქმნას კონკრეტული სისტემები რომლებიც კონკრეტულ დარგზე იქნება მიბმული. კვლევის ერთ-ერთი ძირითადი ამოცანაა პიროვნების იდენტიფიკაცია Wifi ს მეშვეობით. უფრო ზუსტად რომ ვთქვათ: პიროვნება იმყოფება რაიმე შენობაში, სადაც განლაგებულია ოთახები, ხოლო თითოეულ ოთახში კი შესაძლოა იყოს გარკვეული რაოდენობის Wifi როუტერი. აღნიშნულ პიროვნებას გააჩნია მობილური რომელზეც მუდმივად ჩართულია Wifi ზე წვდომა. ჩვენი მიზანია მოვახდინოთ პიროვნების იდენტიფიკაცია ამ წვდომების საფუძველზე. რასაკვირველია ამ ამოცანისთვის შემოდის ისეთი ცნებები როგორიცაა: ტალღის სიხშირე, დამაბრკოლებელი ფაქტორები, ტელეფონისა და როუტერის სპეციფიკა, რომელიც შეიძლება გავლენას ახდენდეს სიხშირეზე და ა.შ. ძირითადი მიზანი არ არის მხოლოდ ამ ამოცანის ამოხსნა, უფრო მთავარს წარმოადგენს გარკვეული სისტემის შექმნა (მანქანური სწავლების საფუძველზე), რომელიც მორგებული იქნება როგორც ამ ამოცანის ამოხსნაზე, ასევე სხვა დარგში არსებულ ამოცანებზე. აუცილებელია შეიქმნას განსხვავებული ტიპის ამოცანებზე მორგებული მოდელი, რომელიც საშუალებას მოგვცემს სხვადასხვა ტიპის ამოცანების გადაჭრა მოვახდინოთ ერთი მოდელის საფუძველზე (რა თქმა უნდა მხოლოდ მოდელი ვერ გადაჭრის ამ ამოცანებს, აუცილებელია მთლიანად ინტელექტუალური სისტემის აგება ამ მოდელის საფუძველზე).

III. კვლევის ობიექტი

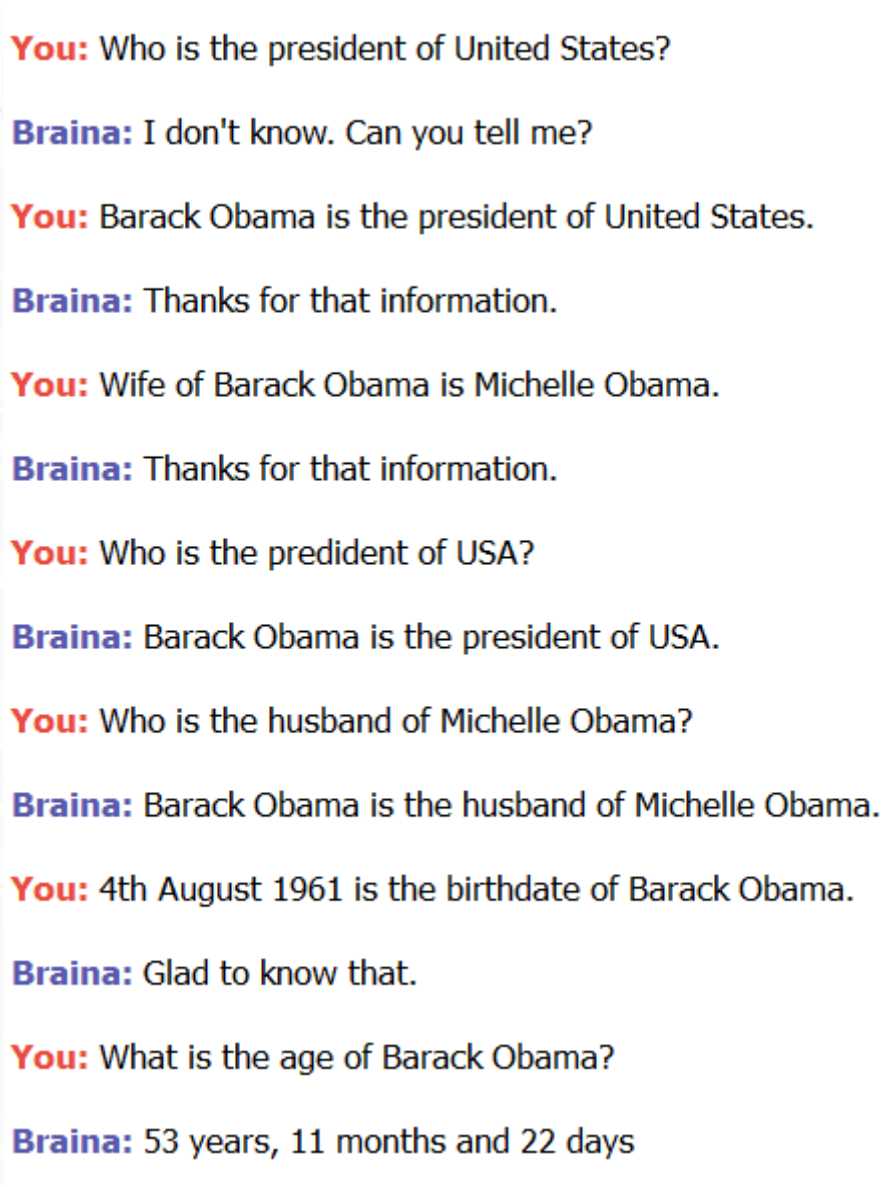
როგორც უკვე აღვნიშნეთ, ინტელექტუალური სისტემა შეიძლება მოიცავდეს უამრავ სხვადასხვა დარგს. კვლევის ობიექტი, როგორც ასეთი, შეიძლება იყოს მეცნიერების სხვადასხვა დარგი, მაგ: მედიცინა, ფიზიკა, ბიოლოგია და ა.შ. ასევე შეიძლება მასში ვიგულისხმოთ კონკრეტული დარგის რაიმე თანამედროვე პრობლემა, მაგ: მედიცინის დარგიდან ონკოლოგიური დაავადებების სწრაფი და დროული დიაგნოსტიკა და ა.შ. საინფორმაციო ტექნოლოგიების კუთხით სულ უფრო და უფრო აქტუალური ხდება ე.წ კიბერ უსაფრთხოების თემები, რაც გულისხმობს შემოჭრების იდენტიფიკაციას, მათ დროულ აღმოჩენასა და აღკვეთას. რასაკვირველია ინტელექტუალურმა სისტემამ უნდა შეძლოს გაარჩიოს ჰაკერული შემოტევა სისტემის ჩვეულებრივი გამოყენებისაგან. იქიდან გამომდინარე, რომ ჰაკერული შემოტევების ტექნოლოგიაც სულ უფრო და უფრო ვითარდება თანამედროვე ტექნოლოგიებთან ერთად, აუცილებელია არსებობდეს ინტელექტუალური სისტემა, რომელშიც თავდაპირველად ჩადებული იქნება ის ძირითადი ბირთვი რაც მიმდინარე მომენტში კაცობრიობას გააჩნია კიბერ შემოტევებთან მიმართებაში. ამის შემდეგ კი აღნიშნულმა სისტემამ თითონ უნდა მოახდინოს განვითარება და მიხვედრა ახალ ახალ შემოტევებზე, სისტემა უნდა მიხვდეს, რომ ეს ახალი ტექნოლოგიით შემოტევაა და ისწავლოს მისი აღკვეთა ადამიანის ჩარევის გარეშე. აღნიშნულიც წარმოადგენს კვლევის ერთ-ერთ უმნიშვნელოვანეს დარგს, რომელიც XXI საუკუნეში საკმაოდ აქტუალურია.

IV. პრაქტიკული მნიშვნელობა

პრაქტიკული მნიშვნელობა გულისხმობს პასუხს ისეთ კითხვებზე როგორიცაა: რას მოგვცემს აღნიშნული ამოცანის გადაჭრა? ხომ არ გამოიწვევს ამ ამოცანების გადაჭრა რაიმე სხვა ახალი ამოცანის წარმოშობას, რომელიც უფრო მეტი სირთულის იქნება? საქმე იმაშია, რომ ზოგიერთი პრობლემა თანამედროვე კაცობრიობის უდიდეს ტკივილად იქცა, აქედან გამომდინარე ასეთი ტიპის ამოცანების გადაჭრა ფრიად მნიშვნელოვანი იქნება. საკმაოდ პროდუქტიული იქნება თუ მართლაც შეიქმნება ერთ მოდელზე დაფუძვნებული ინტელექტუალური სისტემა. ცოტა ხნის წინ გამოვყავით ორი ძირითადი დარგის ამოცანა რომლისთვისაც საკმაოდ ეფექტური იქნება აღნიშნული სისტემის შექმნა. მათგან ერთ-ერთი იყო კიბერ უსაფრთხოების დროული აღკვეთა და მათზე რეაგირება. კიბერ შემოტევის დროს როდესაც სისტემა შეეცდება აღკვეთოს აღნიშნული ინციდენტი, ამას შედეგად შეიძლება მოყვეს განმეორებითი , უფრო პროფესიონალური შემოტევა. ყველაზე ცუდი რაც ასეთ დროს შეიძლება მოხდეს არის ის, რომ შესაძლოა შეიქმნას ინტელექტუალური სისტემა რომელიც არა კიბერ შეტევის აღსაკვეცად გამოიყენება არამედ პირიქით, შეტევისათვის. ამ ყოველივედან გამომდინარე სისტემაში აუცილებლად უნდა იქნას გათვალისწინებული, რომ მანქანური სწავლების შესაბამისი მოდელი (რომელსაც ჩვენ განვიხილავთ) ვერ იქნეს გამოყენებული ბოროტი მიზნებისათვის.

1. არსებული სისტემების ზოგადი მიმოხილვა

ინტელექტუალური სისტემის ერთ-ერთი უმთავრესი მახასიათებელი არის მისი უნარი შეისწავლოს და დაამუშაოს ახალი ინფორმაცია. არსებობს უამრავი სხვადასხვა ცნობილი ინტელექტუალური სისტემები: Braina [3], Mycin [4]. აღნიშნული ინტელექტუალური სისტემები წარმოადგენენ თანამედროვეობის ერთ-ერთ საუკეთესო მიღწევებს. ისინი გარკვეული ფუნქციონალობით გამოირჩევიან. Braina გამოიყენება ოპერაციულ სისტემა Windows ში, რომელიც უზრუნველჰყობს ხმოვანი სიგნალის გარდაქმნას Windows ის ბრძანებებად. ხმოვანი სიგნალი (კონკრეტულად ამ შემთხვევაში) წარმოადგენს ინგლისურად წარმოთქმულ წინადადებებს, რომელსაც შემდეგ სისტემა გარდაქმნის ოპერაციული სისტემის ბრძანებებად. აღნიშნული სისტემა ეფექტურია როდესაც ოპერაციული სისტემის დონეზე გვინდა რაიმე ამოცანების შესრულება. არსებობს Braina ს პროგრამული უზრუნველჰყოფა Android ისთვისაც. მას შეუძლია მოიძიოს ინტერნეტში ინფორმაცია, მოასმენინოს მომხმარებელს ის მუსიკა რაც მას სურს, მოძებნოს ფაილები კომპიუტერში, შეასრულოს სასურველი მათემატიკური ოპერაციები და ა.შ. შემდეგი მაგალითი უკავშირდება Braina ს ხელოვნურ „ტვინს“, მას შეუძლია დიალოგიდან ისწავლოს ახალი ინფორმაცია:



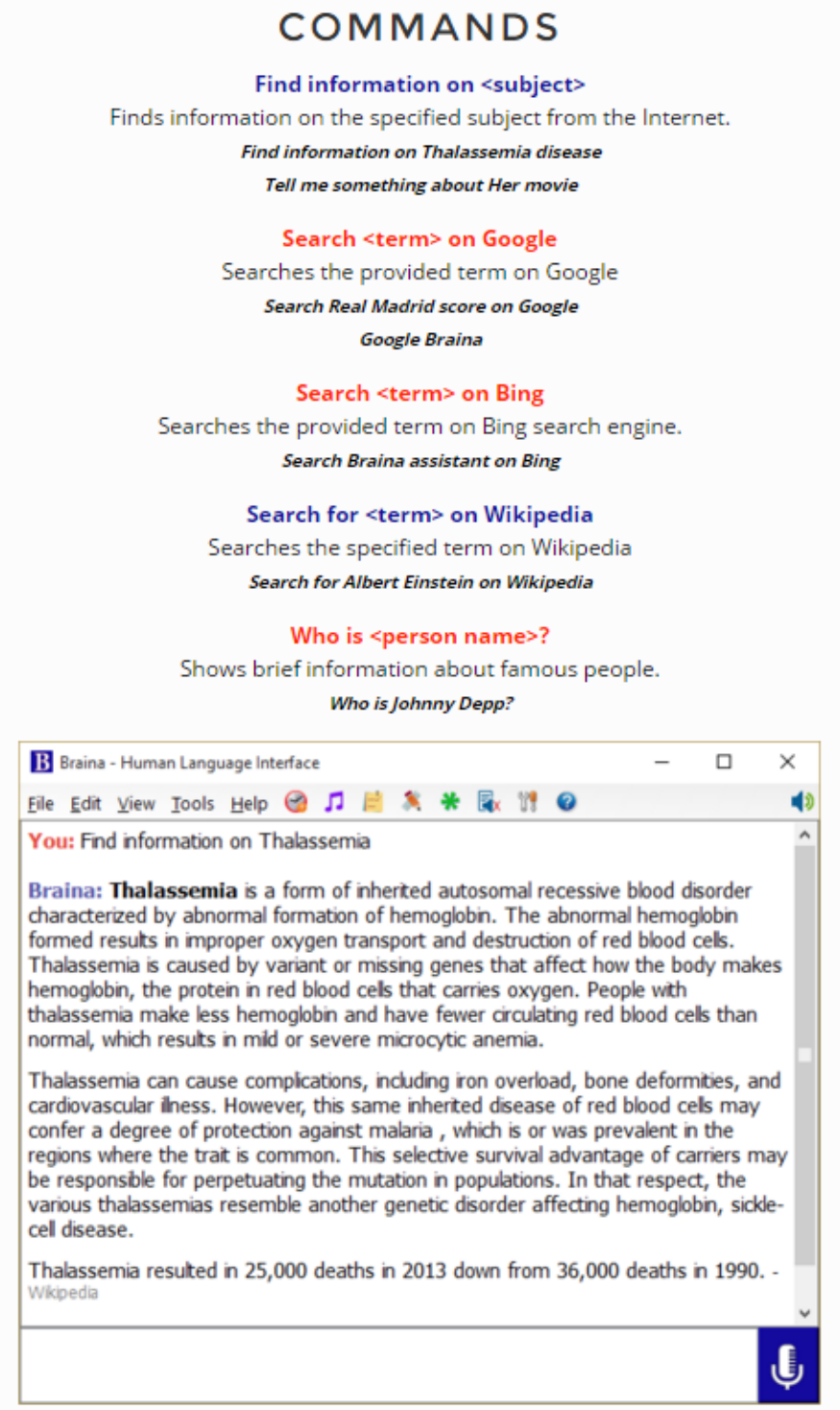
სურ 1. ინფორმაციის დასწავლა

როგორც მაგალითზე ჩანს Braina არა მხოლოდ იმახსოვრებს ახალ ინფორმაციას, არამედ ის ახდენს ამ ინფორმაციის გააზრებულად დალაგებას. აღნიშნულ სისტემას ასევე აქვს უნარი ამოშალოს ინფორმაცია მისი „ტვინიდან“.

სურ. 2 ინფორმაციის ამოშლა

Braina ს ასევე შეუძლია ინფორმაციის დამუშავება ტექსტური ფაილიდანაც. ეს მიდგომა კარგია რადგან სასწავლი ინფორმაცია შეიძლება იყოს დიდი და ადამიანი ამ ინფორმაციას ქმნიდეს გარკვეული პერიოდის განმავლობაში.

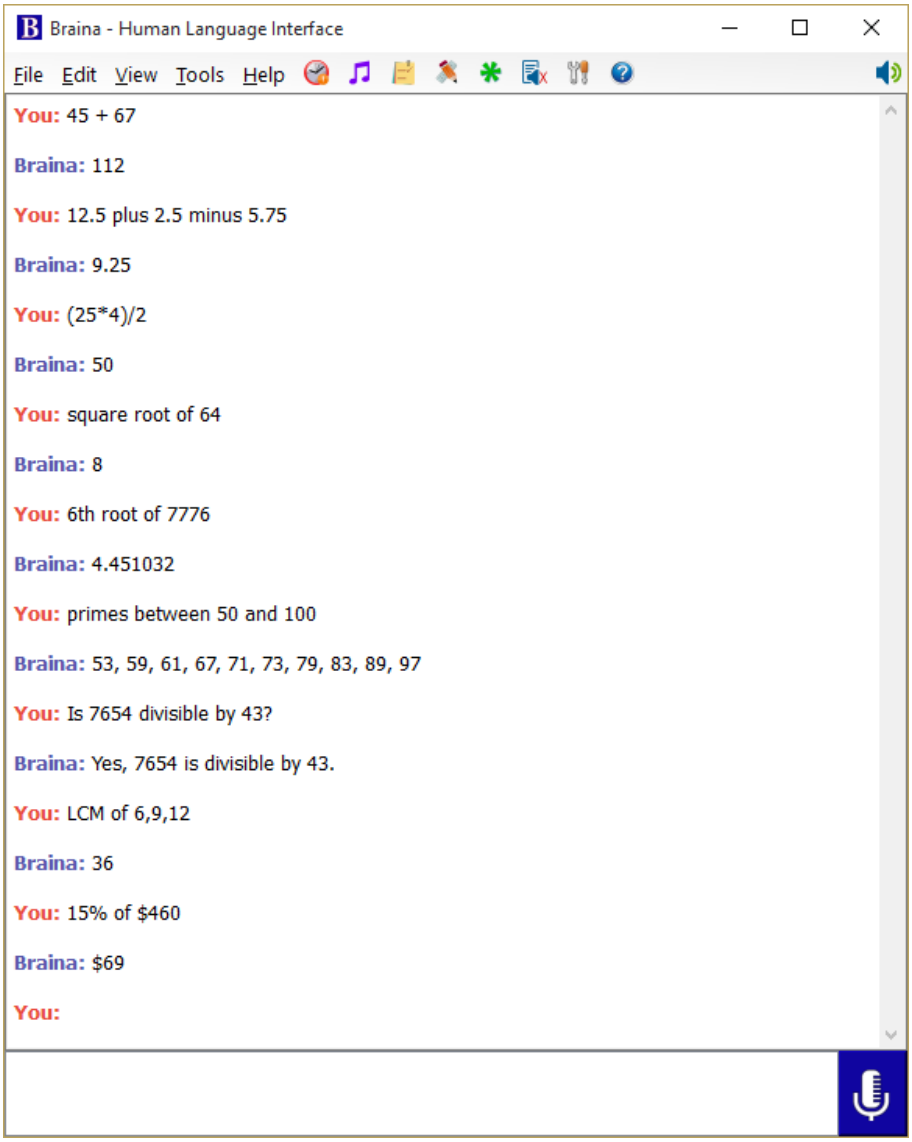
Braina ს ერთ-ერთი მთავარი ფუნქციონალობაა ასევე ბრძანებების საფუძველზე ინტერნეტში ინფორმაციის მოძიება:



სურ. 3 ინტერნეტში ინფორმაციის მოძიება

მის ერთ-ერთ საინტერესო თვისებას ასევე წარმოადგენს მათემატიკური ოპერაციების შესაძლებლობა. თავისი არსით მათემატიკური ოპერაციები შეიძლება იყოს ძალიან კომპლექსური, აქედან გამომდინარე ამ

კერძო შემთხვევაშიც შეიძლება სისტემას დაჭირდეს დასწავლის უნარის გამოყენება:

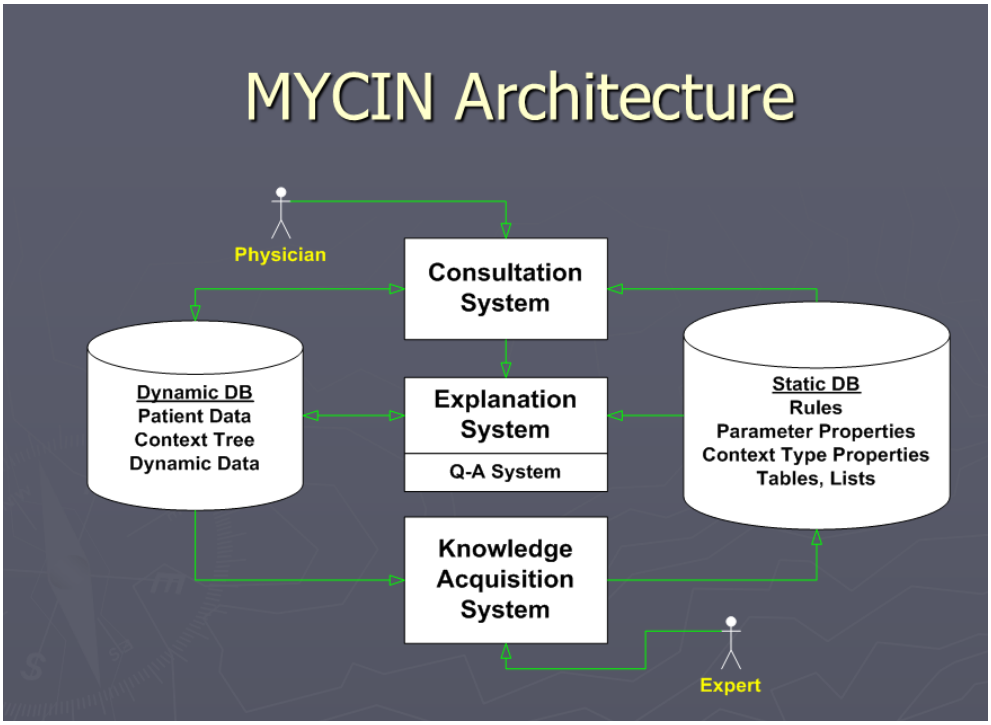


სურ. 4 მათემატიკური ოპერაციები

როგორც დასაწყისში აღვნიშნეთ ინტელექტუალური სისტემის ერთ-ერთი უმთავრესი თვისება არის მისი დასწავლის უნარი (გარედან მიღებული ინფორმაციის საფუძველზე სისტემას უნდა შეეძლოს სწავლა და შემდეგ ამ ცოდნის გამოყენება). ზემოაღნიშნულ მაგალითზე ცხადად არის წარმოდგენილი ის ფაქტი , რომ Braina ამ თვისებას აკმაყოფილებს.

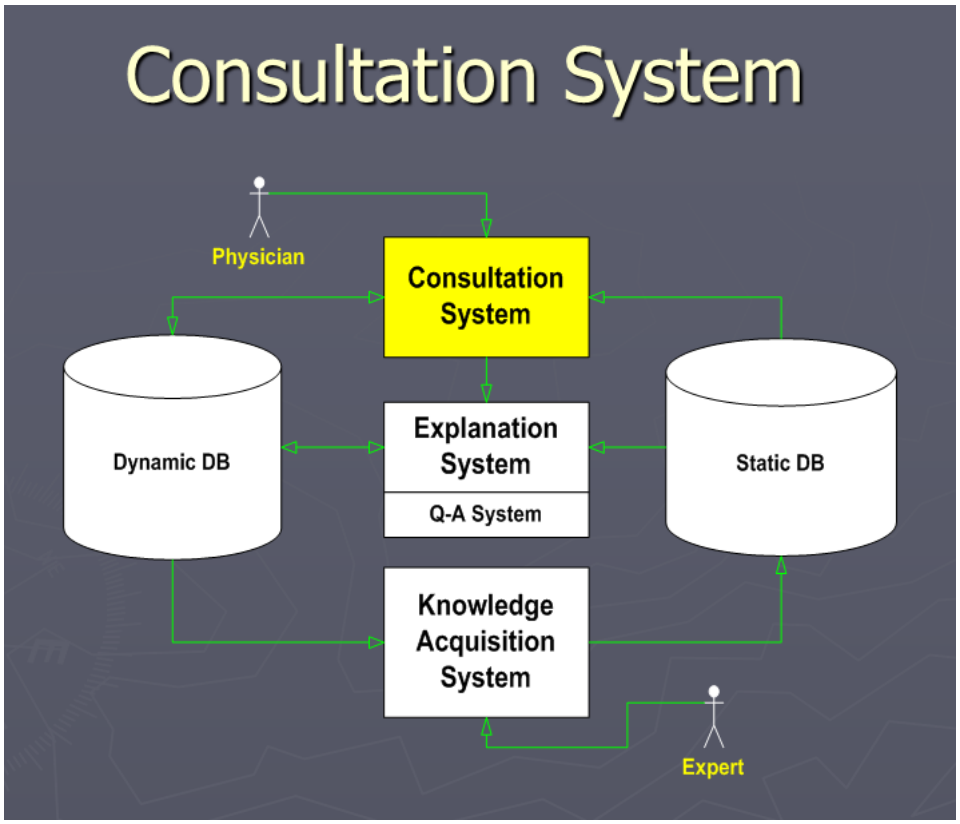
Mycin [16] წარმოადგენს ექსპერტულ სისტემას, რომელიც გამოიყენებოდა ხელოვნურ ინტელექტში ბაქტერიების იდენტიფიკაციისათვის, ისეთი ტიპის დაავადებების დასადგენად,

როგორიცაა მენინგიტი. გარკვეული გადაწყვეტილებების საფუძველზე სისტემა აძლევდა მომხმარებელს რჩევას ანტიბიოტიკების მიღების აუცილებლობაზე (რომლის დოზასაც განსაზღვრავდა პაციენტის სხეულის წონის მიხედვით). აღნიშნული სისტემა ასევე გამოიყენებოდა სისხლის შედედების დიაგნოსტიკისათვის. Mycin ის არქიტექტურა გამოიყურება შემდეგნაირად:



სურ. 5 Mycin ის არქიტექტურა

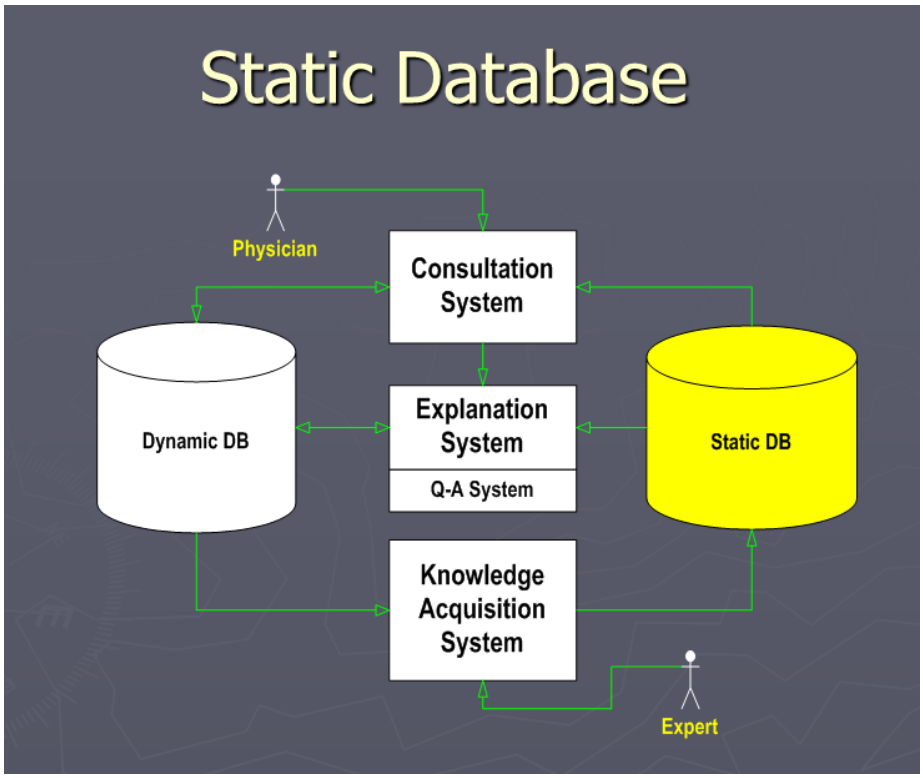
აღნიშნული სისტემის ერთ-ერთ უმთავრეს კომპონენტს წარმოადგენს „საკონსულტაციო სისტემა“, რომელიც არის ერთ-ერთი მთავარი ნაწილი მთლიანი Mycin სისტემის. „საკონსულტაცია სისტემა“ მოიცავს ისეთ ძირითად ფუნქციონალობებს, როგორიცაა: დიაგნოზის დადგენა და თერაპიის განსაზღვრა, ბმულს სხვა ძირითად კომპონენტებთან და ა.შ.



სურ. 6 საკონსულტაციო სისტემის გამოყოფა

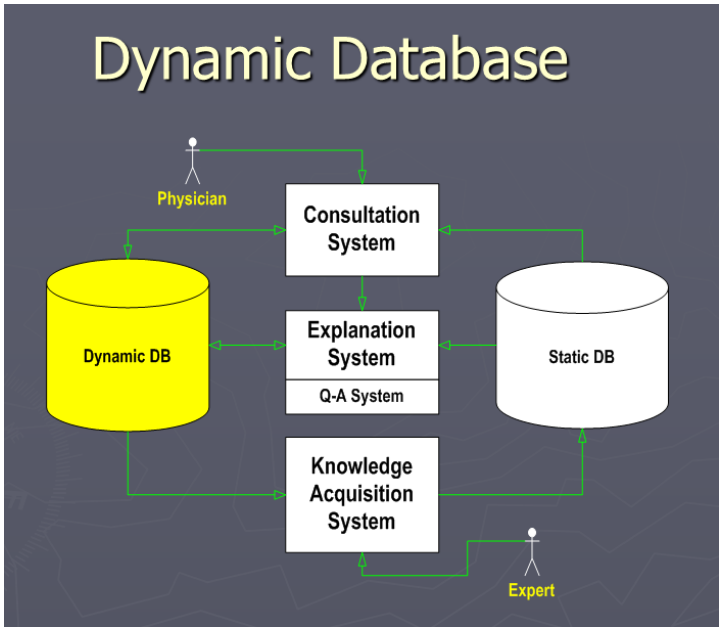
საკონსულტაციო სისტემას გააჩნია კავშირი სხვა ისეთი ტიპის ძირითად კომპონენტებთან, როგორიცაა სტატიკური და დინამიკური ბაზა, ე.წ განმარტებითი სისტემა და ა.შ. ცოდნის ბაზის გარკვეულ დონემდე შევსება ხდება ექსპერტის მიერ.

სტატიკური ბაზა წარმოადგენს ერთ-ერთ მთავარ მახასიათებელს აღნიშნული სისტემის, რომელშიც ინახება ისეთი ძირითადი ინფორმაცია, როგორიც არის: წესები, მეტა წესები, შაბლონები, წესების თვისებები და ა.შ. წესები განსაზღვრავს სისტემის ძირითად ქცევას, ხოლო მეტა წესები კი არის სპეციფიკური ინფორმაცია წესებზე.



სურ. 7 სტატიკური ბაზის გამოყოფა

სტატიკური ბაზის გარდა სისტემას ასევე გააჩნია დინამიური ბაზა, რომელიც მოიცავს ისეთ ძირითად საკითხებს, როგორიცაა: ლაბორატორიული მონაცემები, ინფორმაცია პაციენტზე, განმარტებითი სისტემის გარკვეული რეზულტატები და ა.შ.



სურ. 8 დინამიური ბაზის გამოყოფა

ჩვენ განვიხილეთ რამოდენიმე ინტელექტუალური სისტემა, რომელიც თავისი დანიშნულებით გამოარჩევენ კონკრეტულ დარგებს. ყოველი ასეთი სისტემა იყენებს გარკვეულ ალგორითმებს რათა მიღწეულ იქნას სასურველი შედეგი, თუმცა ასეთი ტიპის სისტემებში გამოყენებული ალგორითმები და მიდგომები შეიძლება განსაზღვრული იყოს მხოლოდ კონკრეტულ სისტემებზე და არ იყოს ზოგადი. აქედან გამომდინარე როდესაც საჭირო გახდება რაიმე ახალი ცოტათი განსხვავებული სისტემის შექმნა, არსებულმა მოდელმა შეიძლება ვერ გადაჭრას ის ძირითადი პრობლემები რომელიც ახალ სისტემაში იქნება. ჩვენი მიზანია შეიქმნას ისეთი მოდელის ბაზისი, რომელიც გამოყენებულ იქნება სხვადასხვა ტიპის ინტელექტუალურ სისტემებში. გამომდინარე იქიდან რომ ინტელექტუალური სისტემები მოიცავენ უამრავ სხვადასხვა დარგებს, მოდელი გათვლილი უნდა იყოს სხვადასხვა ტიპის ინტელექტუალურ სისტემებზე. ჩვენს მიერ შემუშავებულ მოდელში არ იქნება გამოყენებული ერთი კონკრეტული ალგორითმი, არამედ შეიძლება გამოყენებულ იქნას სხვადასხვა ტიპის ალგორითმები, ან/და მოხდეს მათი კომბინაცია. კომბინაცია არ გულისხმობს მხოლოდ ორი ან მეტი ალგორითმის შერწყმას, არამედ ის ძირითადად ეფუძვნება ერთ მოდელში გაერთიანებულ რამოდენიმე სხვადასხვა ალგორითმს. ის ალგორითმები, რომლებიც უნდა გაერთიანდნენ შეიძლება იყოს სხვადასხვა ტიპის და პირდაპირი გზით ერთმანეთთან არ იყოს დაკავშირებული, თუმცა იარსებებს სისტემაში ისეთი გადაბმის წერტილები, რომლებშიც ალგორითმები უშუალოდ უნდა ვალაპარაკოთ ერთმანეთთან. ამ ეტაპზე უკვე საქმეში ერთვება მონაცემთა ნორმალიზაცია/დენორმალიზაცია, რომლის საშუალებითაც ვახდენთ ინფორმაციის ფორმირებას. ნორმალიზაცია/დენორმალიზაცია საჭიროა მთლიან სისტემაში თუმცა არის აუცილებელი წერტილები, სადაც ამ საკითხს გვერდს ვერ ავუვლით, კერძოდ:

- **ნორმალიზაცია საწყის ეტაპზე** - სანამ სისტემა დაიწყებს უშუალოდ მონაცემებთან მუშაობას (მითუმეტეს როდესაც ეს

მონაცემები არის გარედან შეყვანილი) აუცილებელია მათი გარდაქმნა სისტემისათვის გასაგებ ენაზე.

- **ნორმალიზაცია სისტემის მუშაობის ეტაპზე** - როდესაც სისტემა მუშაობს (ეს ასევე ნიშნავს იმასაც რომ გარკვეული ალგორითმები ერთმანეთში ცვლიან ინფორმაციას) აუცილებელია გავითვალისწინოთ, რომ ერთი ალგორითმის შედეგი (გამომავალი ინფორმაცია) აუცილებლად იქნება გამოყენებული როგორც მეორე ალგორითმის შემავალი ინფორმაცია (თუმცა შეიძლება ერთზე მეტი ალგორითმი იყენებდეს აღნიშნულ გამომავალ ინფორმაციას). აქედან გამომდინარე აუცილებელია მონაცემების გარდაქმნა პროცესში არსებული შემდეგი ალგორითმის გასაგებ ენაზე.
- **დენორმალიზაცია სისტემის მუშაობის საბოლოო ეტაპზე** - როდესაც სისტემა მოახდენს ამოცანის შესრულებას, საბოლოო ჯამში აუცილებელია გამომავალი ინფორმაცია, რომელსაც ადამიანი ადექვატურად აღიქვამს. აქედან გამომდინარე საჭირო ხდება საბოლოო ინფორმაციის ისეთ ფორმატში წარმოდგენა, რომელიც გასაგები იქნება შესაბამისი სისტემის მომხმარებლისთვის.

ზემოთ თქმულიდან გამომდინარე აღნიშნული მიდგომა ძირითად მოიცავს შემდეგ ძირითად ნაწილებს: ინფორმაციის ნორმალიზაცია/დენორმალიზაცი, ალგორითმები რომლებიც შეიძლება იყოს სხვადასხვა ტიპის, ამ ალგორითმების კომბინაცია (გარკვეული მიმდევრობით).

2. ერთ-ერთი ძირითადი ამოცანა

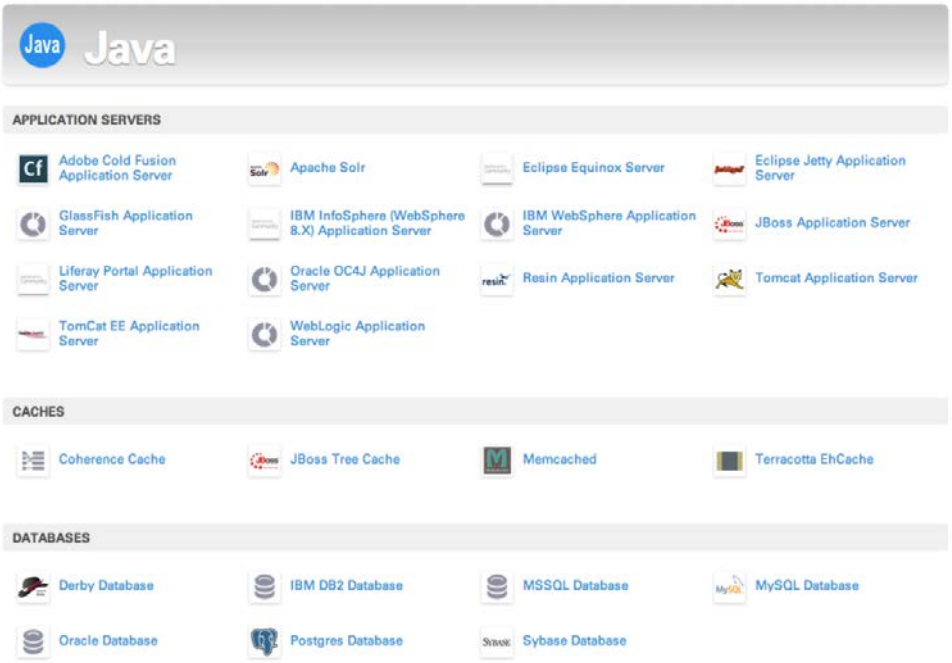
თანამედროვე მსოფლიოში არსებობს უამრავი ისეთი ამოცანა, რომელიც ან არ არის გადაჭრილი ან გადაწყვეტილია რაიმე არა ეფექტური გზით (ამის მიზეზი შეიძლება მრავალნაირი იყოს, მაგ: კაცობრიობის ცოდნის დონე კონკრეტულ დარგში და ა.შ). თუმცა წარსულშიც იყო ისეთი ტიპის პრობლემები, რომლებიც ერთი შეხედვით გადაუჭრელი ჩანდა და სანამ კაცობრიობამ ამ პრობლემის გადაჭრის გზა იპოვა უამრავი ადამიანი დაიღუპა. ასეთი ტიპის პრობლემები განსაკუთრებით ბევრი იყო მედიცინაში (ახლაც არ არის ცოტა თუმცა დაავადებების ნაირსახეობებიდან გამომდინარე მათი სპეციფიკაცია და აქედან გამომდინარე პრობლემების რაოდენობა იცვლება).

მედიცინია რასაკვირველია არ არის ერთად ერთი დარგი სადაც არსებობს დარგობრივი გადაუჭრელი პრობლემები (უბრალოდ ნებისმიერი ასეთი ტიპის პრობლემები კაცობრიობაზე პირდაპირი გზით აისახება). ინფორმაციული ტექნოლოგიებიც მიეკუთვნება ისეთ დარგს სადაც საკმაოდ მწვავედ დგას ზოგიერთი პრობლემა. ერთ-ერთ ასეთ პრობლემად მოისაზრება უსაფრთხოება (კიბერ უსაფრთხოება). თანამედროვე სამყაროსათვის ცნობილია, რომ რაც დრო გადის უფრო და უფრო აქტუალური ხდება კიბერ უსაფრთხოების ტიპის პრობლემები. აქ ჩნდება უამრავი ისეთი ტიპის კითხვა, როგორიცაა: როგორ დავწერო სისტემა/პროგრამა ისე, რომ ვერ მოხერხდეს მისი გატეხვა? შეგვიძლია ეს კითხვა დავსვათ ცოტა სხვა კუთხითაც: როგორ დავწერო პროგრამა ისეთნაირად , რომ ვერ მოხდეს მისი ოფიციალურად ნაყიდი ლიცენზიის გარეშე (ე.წ კრეკების გარეშე) მისი გაშვება? რასაკვირველია ამ კითხვებზე პირდაპირი პასუხი არ არსებობს, თუმცა არსებობს ფაქტი, რომლის ჭეშმარიტებაც დღევანდელ მსოფლიოში დასტურდება: **ნებისმიერი პროგრამა/სისტემა გატეხვადია, მაგრამ გააჩნია რა დროში**. იქიდან გამომდინარე, რომ არსებულ ფაქტებზე დაყრდნობით შეუძლებელია შეიქმნას გაუტეხავი სისტემა ან თუნდაც უსაფრთხოების მოდელი,

პრობლემის გადაჭრა უფრო მეტად შეიძლება ვცადოთ სხვანაირი გზით: შეიძლება შეიქმნას სისტემა, რომელიც იქნება ცალკე მდგომი და რომელიც მონიტორინგს გაუკეთებს დასაცავ სისტემებს. აღნიშნული ინტელექტუალური სისტემა პირველ ჯერზე განსაზღვრავს რაიმე საბაზისო ცოდნას, ხოლო შემდეგ თვითონ მოახდენს განვითარებას. ეს ნიშნავს იმას, რომ როდესაც სისტემაში მოხდება რაიმე სახის შემოჭრა (ან შემოჭრის მცდელობა), სისტემა თვითონ გაუკეთებს ანალიზს ამ ყოველივეს და მოახდენს ისეთნაირად გადაწყობას, რომლის მიხედვითაც მოხდება დაცვა/შემოტევის აღკვეთა. არსებობს რამოდენიმე სახის სისტემა/პროგრამული უზრუნველჰყოფა, რომლის საშუალებითაც ხდება N დონიანი არქიტექტურის მქონე აპლიკაციების მონიტორინგი (ფართოდ არის გავრცელებული 3 დონიანი არქიტექტურის პროგრამული უზრუნველჰყოფა, რომელიც განსაზღვრულია როგორც **კლიენტი -> სერვერი -> ბაზა** ტიპის სისტემა), სუსტი წერტილის დადგენა. ერთ-ერთი ასეთი ტიპის პროგრამული უზრუნველჰყოფაა App Dynamics[5]. მისი საშუალებით ხდება აპლიკაციის დონეზე უამრავი სხვადასხვა ტიპის მონიტორინგი. მოიცავს სხვადასხვა პროგრამირების ენებს:

სურ. 9 - სხვადასხვა დაპრ.ენების და ტექნოლოგიების მხარდაჭერა

რასაკვირველია თითოეული პროგრამირების ენა შეიძლება მოიცავდეს სრულიად განსხვავებული ტიპის ტექნოლოგიებს (ასევე შეიძლება ჰქონდეს ბევრი რამ საერთოც). პროგრამირების ენა Java ზე, გვაქვს შემდეგი სერვერევის, ტექნოლოგიებისა და მონაცემთა ბაზების მხარდაჭერები:



სურ. 10 სერვერების, ტექნოლოგიებისა და ბაზების მხარდაჭერა დაპრ.ენა Java ზე

თითოეული პროგრამირების ენა შეიძლება მოიცავდეს სხვადასხვა ტექნოლოგიებს. როდესაც სისტემა საკმაოდ დიდია, მის წარმოსადგენად შეიძლება გამოყენებულ იქნას გრაფის მოდელი, რომლის საშუალებითაც მთლიანი სისტემის ვიზუალიზაცია გაცილებით მარტივია. AppDynamics ი იყენებს ამ მოდელს მთლიანი სისტემის ვიზუალიცაზიისათვის, ასევე ამონიტორინგებს თითოეულ კვანძს და აქედან გამომდინარე საშუალებას იძლევა დაკვირვების შედეგად გამოვლინდეს ის სუსტი წერტილები, რომელიც აღმოჩენილ იქნა უშუალოდ სისტემის მუშაობის პროცესში. AppDynaminc ს შეუძლია დაკვირვება პროტოკოლების (SOAP [17], HTTP [18] და ა.შ), სერვერების (Jboss [19], Apache [20] და ა.შ) SQL [21] ისა და სხვა ტექნოლოგიების დონეზე, შესაბამისად თუკი სისტემის ამ დონეზე წარმოიშვა რაიმე სახის პრობლემა, შესაძლებელია ადმინისტრატორს/შესაბამის პასუხისმგებელ პირს ავტომატურად ეცნობოს ამ პრობლემის შესახებ.

სურ. 11 სისტემის წარმოდგენა გრაფიკულად

აღნიშნული სისტემა საკმაოდ პოპულარულია გამოყენების თვალსაზრისით, თუმცა ის გამოიყენება მხოლოდ მონიტორინგისათვის, მას არ შეუძლია შესთავაზოს ამა თუ იმ პრობლემის წარმოშობის დროს რაიმე კონკრეტული გადაწყვეტილება. ჩვენს მიერ შემუშავებულ მოდელს უნდა შეეძლოს არა მხოლოდ სისუსტის დადგენა, არამედ რაიმე გადაწყვეტილების მიღებაც (გადაწყვეტილებას შესაძლოა გააჩნდეს რამოდენიმე ალტერნატივა, ასეთ შემთხვევაში სისტემამ უნდა შესთავაზოს მომხმარებელს ყველა შესაძლო გადაწყვეტილება და ადამიანმა უნდა აირჩიოს ამ გადაწყვეტილებებიდან ერთ-ერთი). ინტელექტუალური სისტემის ერთ-ერთი განმასხვავებელი ფაქტორიც ხომ ისაა, რომ მას უნდა შეეძლოს ახალი ინფორმაციის საფუძველზე სწავლა და თვითგანვითარება, ასევე გარკვეული მზა გადაწყვეტილების მიღება/შემოთავაზება.

მაგალითად შეგვიძლია მოვიყვანოთ ერთი საინტერესო ამოცანა, რომლის გადაწყვეტის გზაც შეგვიძლია ვიპოვოთ ჩვენს მიერ შემუშავებული მოდელის საფუძველზე (რასაკვირველია შესაძლოა არსებობდეს ამ ამოცანის გადაწყვეტის უფრო ეფექტური საშუალებები, თუმცა ჩვენს მიერ შემუშავებული მოდელი, რომელსაც მოგვიანებით აღვწერთ არის ზოგადი და გამოგვადგება სხვადასხვა ტიპის/დარგის ამოცანების გადასაჭრელად).

მოცემული გვაქვს გარემო სადაც განთავსებულია რამოდენიმე სართულიანი შენობა. აღნიშნული შენობა თავისი არქიტექტურით შეიძლება იყოს მრავალნარი და მოიცავდეს უამრავ სხვადასხვა ოთახს. თითოეული ოთახი სხვადასხვა მანძილით არის დაცილებული ერთმანეთს. შენობაში სხვადასხვა ოთახებთან (ან ოთახში) განთავსებულია Wifi მოწყობილობები (ცალკე საკითხია თითოეული ეს მოწყობილობა რა მანძილზე მუშაობს კარგად). შენობაში იმყოფება პიროვნება, რომელსაც თან აქვს მობილური მოწყობილობა. აღნიშნულ პიროვნებას მუდმივად ჩართული აქვს Wifi მის მობილურზე. ის მოძრაობს ოთახიდან ოთახში, სართულიდან სართულზე, ასევე გადის და შემოდის შენობაში. საბოლოო მიზანია ნებისმიერ ეტაპზე დავადგინოთ თუ რომელ ოთახში შეიძლება იმყოფებოდეს პიროვნება. აღნიშნულ ამოცანას შეგვიძლია ვუწოდოთ **Wifi ს მეშვეობით პიროვნების იდენტიფიკაციის ამოცანა.**

ეს ამოცანა შეიძლება ემყარებოდეს უამრავ სხვადასხვა დეტალს, რომლის არსებობამაც შესაძლოა გაართულოს მისი ამოხსნა. შესაძლოა დაგვჭირდეს ისეთ კითხვებზე პასუხის გაცემა როგორიცაა: რა მანძილითაა დაცილებული ერთი ოთახი მეორეს? თუკი ეს ამოცანა უნდა გადაიჭრას მრავალსართულიანი მოდელის მაგალითზე როგორ მოვახდინოთ ოთახების იდენტიფიკაცია თუ ერთი ოთახისთვის ორი ან მეტი ოთახი თანაბრადაა დაცილებული? რა მასალისგან არის დამზადებული კედლები და რას წარმოადგენს ის დამაბრკოლებელი ფაქტორები, რომელთაც შესაძლოა გამოიწვიონ სიგნალის შესუსტება ან სრულიან ჩახშობა? ასეთი ტიპის კითხვები შეიძლება იყოს ბევრნაირი რაც რასაკვირველია ართულებს ამოცანას. ჩვენი მიზანი ამ ეტაპზე არ არის კონკრეტულად ამ ამოცანის სრულყოფილად გადაჭრა, არამედ საჭიროა წარმოვადგინოთ ზოგადი მოდელი (როგორც უკვე აღვნიშნეთ), რომელიც უბრალოდ დაგვეხმარება აღნიშნული ამოცანის გადაწყვეტაში. გამორიცხული არ არის, რომ ამოცანის პირობის მკვეთრად გართულებამ გამოიწვიოს მოდელში გარკვეული ცვლილებების შეტანა ან მთლიანი მოდელის გადაკეთება.

უფრო უკეთესი ვიზუალიზაციისათვის შეგვიძლია მოვიყვანოთ ერთ ერთი სართულის ჭრილი, სადაც კარგად ჩანს ოთახების განლაგება:

სურ. 12 ერთ-ერთი სართულის ჭრილი, სადაც ოთახების განლაგება ჩანს

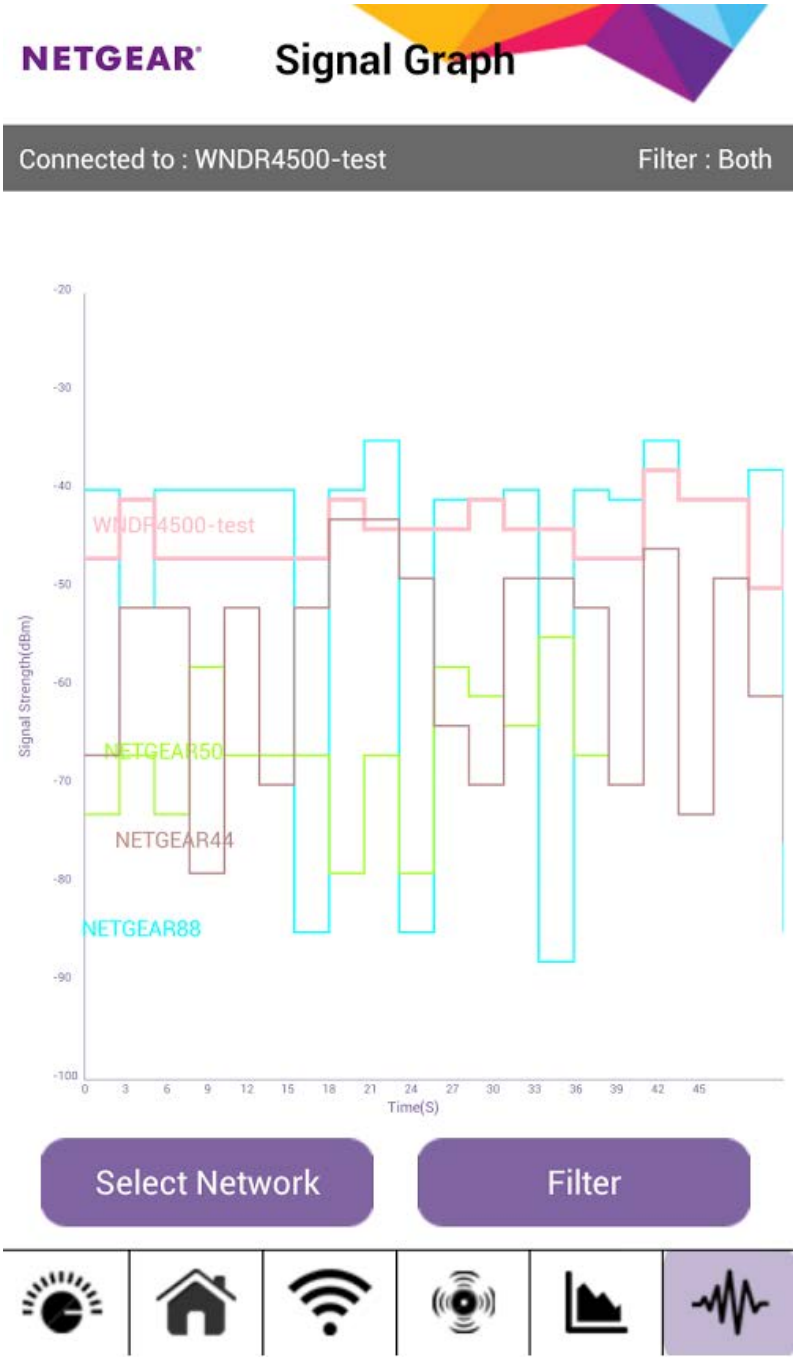
მოცემულ სიტუაციაში შესაძლოა Wifi მოწყობილობები სხვადასხვა სახით იყოს განლაგებული, შესაბამისად საინტერესოა იმის დადგენაც, როგორ იცვლება მოდელის სიზუსტე Wifi მოწყობილობების განლაგების ცვლილებასთან ერთად. ამის საფუძველზე მოდელისათვის შესაძლოა განხილულ იქნას სხვადასხვა შესაძლო შემთხვევები: იდეალური შემთხვევა (Wifi მოწყობილობები ისეა განლაგებული, რომ აღნიშნულ მოდელს ერგება იდეალურად), ცუდი შემთხვევა (Wifi მოწყობილობები ისეა განლაგებული, რომ აღნიშნული მოდელი ვერ პოულობს გადაწყვეტილებას ან მის მიერ ნაპოვნი გადაწყვეტილება არ არის ზუსტი), საშუალო შემთხვევა.

ერთ-ერთი მნიშვნელოვანი საკითხი, რაც ამოცანის განუყოფელი ნაწილია არის Wifi სიგნალები, რომელსაც გააჩნია სიხშირე. ერთი შეხედვით შეგვიძლია დავეყრდნოთ მხოლოდ სიხშირეებს და მის საფუძველზე ვთქვათ რომელ ოთახთან იმყოფება პიროვნება, მაგრამ ასეთ შემთხვევაში შეიძლება მოხდეს ისე, რომ პიროვნება იმყოფებოდეს ისეთ ადგილას საიდანაც ორი ან მეტი Wifi მოწყობილობა თანაბრად არის

დაცილებული? ან ერთი მოწყობილობა უკეთესი სიხშირით გამოირჩევა, ხოლო მეორე უფრო სუსტი სიხშირით. ეს და სხვა ასეთი ტიპის ფაქტორი ხელს გვიშლის გადავჭრათ ეს ამოცანა მხოლოდ სიხშირეზე დაყრდნობით თუმცა ამ თვისებას გვერდს ვერ ავუვლით. არსებობს უამრავი სხვადასხვა პროგრამული უზრუნველჰყოფა, რომლის საშუალებითაც ხდება Wifi სიგნალების გაზომვა. ასეთი ტიპის პროგრამები უმეტესად მობილური მოწყობილობებისთვის არის განსაზღვრული. ერთ-ერთი ასეთი პროგრამაა Wifi Analytics [6] , რომელსაც გააჩნია სხვადასხვა ფუნქციონალი. ერთ-ერთი ასეთი ფუნქციონალია სიგნალის გაზომვა (მიერთებულ Wifi ზე):

სურ. 13 Wifi სიგნალის გაზომვა სპეციალური პროგრამით

აღნიშნულ პროგრამას ასევე აქვს შესაძლებლობა აქვს ააგოს სიგნალების გრაფი:



სურ. 14 სიგნალების გრაფის ჩვენება სპეციალური პროგრამით

სანამ პროგრამა მოახდენს უშუალოდ გრაფის აგებას, აუცილებელია მოვნიშნოთ ყველა ის Wifi მოწყობილობა (ან მხოლოდ ის მოწყობილობები, რომელთან კავშირიც არის საინტერესო), რომელსაც მიმდინარე მომენტში შეგვიძლია მივუერთდეთ.

არის კიდევ ერთი დეტალი, რომელზეც შეგვიძლია გავამახვილოთ ყურადღება: სხვადასხვა მობილური მოწყობილობები გამოირჩევიან როგორც სხვადასხვა ტიპის ოპერაციული სისტემებით (ძირითადად ესენია Android[23], IOS[24], BlackBerry[25]) და შესაბამისი პროგრამული უზრუნველჰყოფით, ასევე სხვადასხვა აპარატურით (ფიზიკური ნაწილებით). მხოლოდ Wifi მოწყობილობის სიგნალის სიხშირეზე დაყრდნობა შეიძლება არ იძლეოდეს ადექვატურ შედეგს, აქედან გამომდინარე საჭირო ხდება იმის გადამოწმებაც თუ რამდენად კარგად აღიქვავს თავად მობილური მოწყობილობა გარე Wifi სიგნალებს. ასევე მნიშვნელოვანი განვსაზღვროთ გარკვეული კრიტერიუმები, რომლის მიხედვითაც გავაკეთებთ დასკვნას სიგნალის შესახებ, კარგია აღნიშნული სიგნალი თუ ცუდი:

სურ. 15 ცუდი და კარგი სიგნალის სიხშირეები

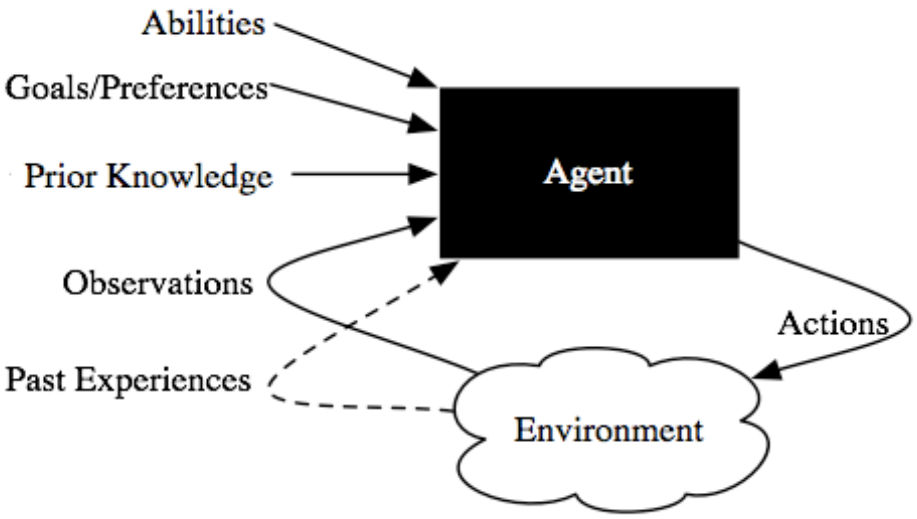
აღნიშნული შეგვიძლია განვსაზღვროთ მიმართებებით:

RSSI	Signal Strength
> -70 dBm	Excellent
-70 dBm to -85 dBm	Good
-86 dBm to -100 dBm	Fair
< -100 dBm	Poor
-110 dBm	No signal

სურ. 16 Wifi სიგნალების ხარისხის განსაზღვრა მიმართებებით

სიხშირის სიცუდის და სიკარგის განმსაზღვრელი კრიტერიუმები შესაძლოა შეიცვალოს ამოცანის დასმასთან ერთად თუმცა ზემოაღნიშნული წარმოადგენს კონკრეტულ მაგალითს მხოლოდ თუ როგორ შეგვიძლია გამოვყოთ ერთმანეთისაგან კარგი და ცუდი ტიპის სიგნალები.

ინტელექტუალური სისტემა (როგორიც არ უნდა იყოს ის) წარმოუდგენელია ე.წ ინტელექტუალური აგენტის გარეშე. არსებობენ სხვადასხვა ტიპის აგენტები, მაგ: აგენტები, რომლებიც პრობელემებს წყვეტენ ძიების მეშვეობით, დასწავლადი აგენტები და ა.შ. აგენტებს ახასიათებთ მოქმედების უნარი, შესაბამისად, ნებისმიერი პროგრამა რომელიც გარკვეულ გააზრებულ ქმედებებს ასრულებს შეიძლება ჩაითვალოს ინტელექტუალურ აგენტად.



სურ. 17 აგენტი და მისი შესაძლო თვისებები

მოკლედ შეგვიძლია დავახასიათოთ მისი ზოგიერთი თვისება:

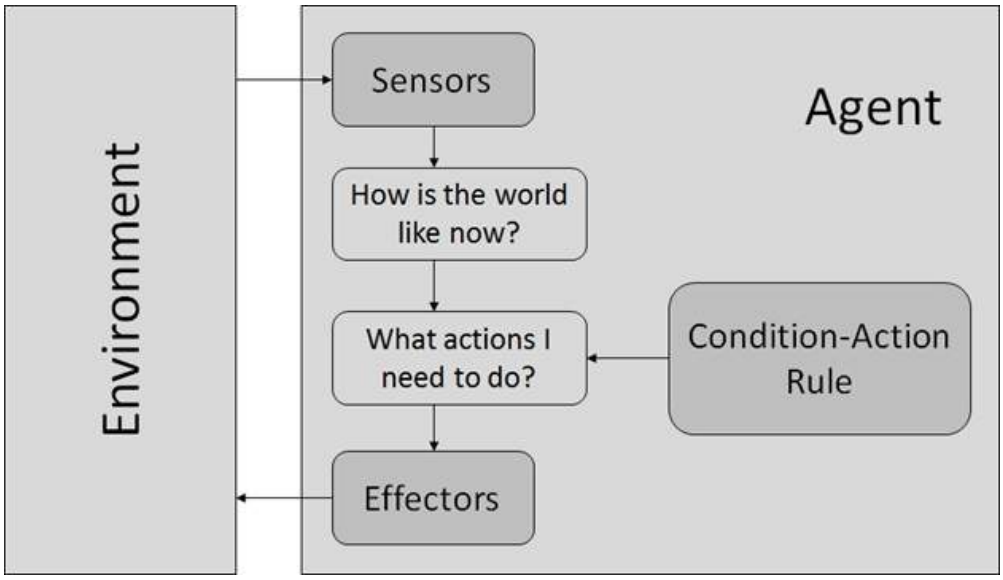
- **Abilities** - წარმოადგენს იმ ინტელექტუალურ თუ ფიზიკურ შესაძლებლობებს, რაც შეუძლია აგენტს განახორციელოს, გამოიყენოს;
- **Goals** - წარმოადგენს რაიმე დასახულ მიზანს. ეს მიზანი შეიძლება იყოს ლოკალური (კერძო ბიჯზე მისაღწევი) და გლობალური (ამოცანის საბოლოო მიზანი);
- **Prior knowledge** - ცოდნა/დალაგებული ინფორმაცია, რომელიც გააჩნია აგენტს მიმდინარე მომენტში. რასაკვირველია ეს ცოდნა არის დინამიური, გარკვეულ მომენტში (ტექნიკური მეხსიერებიდან გამომდინარე) აგენტის მეხსიერებაში იზრდება ინფორმაციის რაოდენობა, საჭიროებისამებრ ხდება ძველი, არასაჭირო ინფორმაციის წაშლა;
- **Observations** - წარმოადგენს გარკვეულ დაკვირვებებს, რომელსაც აგენტი ახორციელებს გარემოზე (ამის კლასიკური მაგალითია ავტომატურად მართვადი მანქანა [1], რომელსაც მძღოლის გარეშე შეუძლია აღიქვას ობიექტები, მოახდინოს მათი იდენტიფიკაცია და ამის საფუძველზე მიიღოს გარკვეული გადაწყვეტილება);

- **Actions** - შესაძლო ქმედებები, რომელსაც აგენტი ახორციელებს საჭირო დროს. ჩვენი ამოცანის შემთხვევაში ასეთი ქმედებები იქნება გარკვეული ალგორითმის გამოძახება, კონკრეტული გადაწყვეტილების მიღება და ა.შ.
- **Past Experiences** - წარმოადგენს წარსულში დაგროვილ გარკვეულ ინფორმაციას, რომელსაც აგენტი იყენებს გარემოსთან/პრობლემასთან მიმართებაში. წარსული გამოცდილება ეხმარება აგენტს ამოცანის სწრაფ გადაწყვეტაში ამდენად ეს პარამეტრი წარმოადგენს ერთ-ერთ მნიშვნელოვან ფაქტორს საბოლოო მიზნის მისაღწევად;

აგენტი მოქმედებს გარკვეულ გარემოში. გარემო კი შეიძლება იყოს ორი სახის:

- **დეტერმინისტული** - წარმოადგენს ისეთი ტიპის გარემოს რომელიც შემოსაზღვრულია და ამ გარემოდან გასვლა შეუძლებელია. ამის მაგალითია ჭადრაკის დაფა, როდესაც ორი ადამიანი თამაშობს ამ თამაშს, როგორც არ უნდა ითამაშონ (რა სვლებიც არ უნდა მოიფიქრონ მოთამაშეებმა) ვერ გაცდებიან ამ გარემოს.
- **სტოქასტური** - წარმოადგენს დინამიურ გარემოს, რომელიც არ არის შემოსაზღვრული რაიმე წინასწარგანსაზღვრული საზღვრებით.

ინტელექტუალური სისტემა ერთიანობაში მოიცავს სხვადასხვა კომპონენტებს. ეს ყოველივე სქემატურად შეგვიძლია ასეთნაირად წარმოვადგინოთ:



სურ. 18 ინტელექტუალური სისტემა, როგორც ერთიანი სისტემა

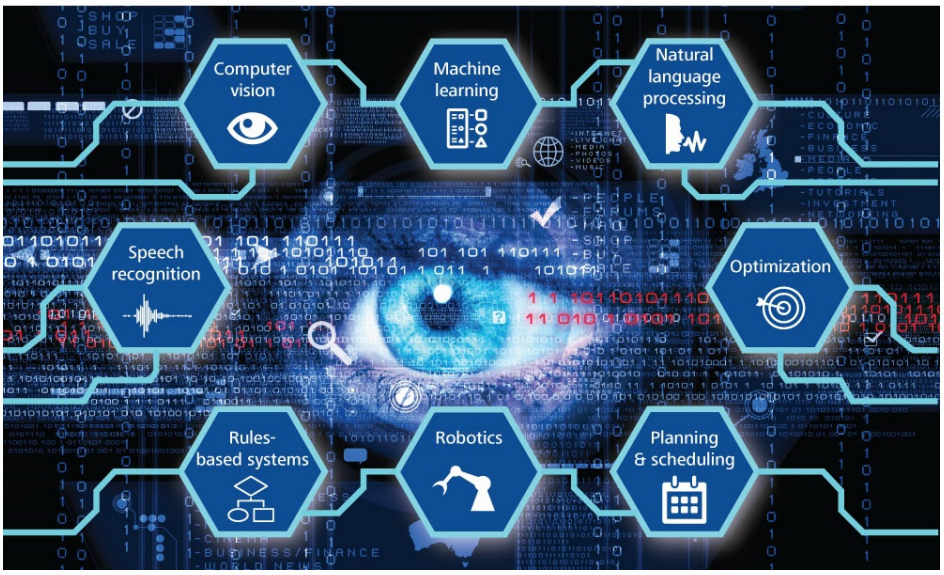
როგორც აღვნიშნეთ ნებისმიერ ამოცანას ახასიათებს გარკვეული გარემო. გარემოს კი ახასიათებს გარკვეული გაუთვალისწინებელი ფაქტორები (მაგ: სტიქიური უბედურებები და ა.შ). ერთი საკითხია ტექნიკურად როგორ გადაიჭრება ესა თუ ის ამოცანა ხოლო მეორე არანაკლებ მნიშვნელოვან ფაქტორს წარმოადგენს გარკვეული გარემო პირობები და გაუთვალისწინებელი შემთხვევები, რომელთა წინაშე ადამიანიც კი შესაძლოა უძლური აღმოჩნდეს. ეს ფაქტი არ გულისხმობს პროგრამის მიერ გადაუჭრელ პრობლემებთან შეჯახებას ან/და რაიმე მცდელობას რამენაირად წინ აღუდგეს ისეთ მოვლენებს რომელმაც თუდაც შესაძლოა ნგრევა და ფიზიკური განადგურება გამოიწვიოს, არამედ ძირითადად უნდა უზრუნველჰყოფდეს დამუშავებული ინფორმაციის იმგვარად შენახავას, რომლის აღდგენაც და პროცესის გაგრძელებაც პოტენციურად შესაძლებელი იქნება ასეთი მოვლენის შემდეგაც. რასაკვირველია ამ უკანასკნელის გათვალისწინება არც თუ ისე იოლია და ეს ყოველივე მოითხოვს არა მხოლოდ კარგ პროგრამულ უზრუნველჰყოფას არამედ კარგ აპარატურას და მრავალი ფიზიკური ფაქტორის გათვალისწინებას.

ჩვენს მიერ განხილულ ამოცანაში გარემო არის შემდეგი:

- **შენობა** - რომელიც არის განთავსებული კონკრეტულ ფიზიკურ ადგილას და რომელსაც გააჩნია ფიზიკური მისამართი;
- **ნებისმიერი ბუნებრივი მოვლენა** - მისი შედეგი შესაძლოა პირდაპირ აისახოს აღნიშნული შენობის და მასში მყოფი ადამიანების კეთილდღეობაზე. ჩვენი ამოცანის შემთხვევაში თუ დავუშვებთ იმას, რომ მოხდა მაგალითად მიწისძვრა, რომელსაც შეუძლია სრულიად გაანადგუროს აღნიშნული შენობა, მაშინ არანაირი აზრი არ ექნება იმას თუ რამდენად კარგად იმუშავებს პროგრამული უზრუნველჰყოფა.
- **ადამიანის მიერ კონტროლირებადი ფაქტორები** - ჩვენს მაგალითში ასეთს წარმოადგენს ელექტროენერგია, რომელიც შესაძლოა ხელოვნურად იქნას გათიშული (თუმცა აქაც საქმეში ერთვება გარემო ფაქტორები, რადგან ეს გათიშვა შეიძლება გამოწვეული იყოს არა ადამიანის სუბიექტური სურვილიდან გამომდინარე, არამედ გარკვეული დაზიანებების საფუძველზე, რაც წარმოადგენს ობიექტურ მიზეზს).

აქედან გამომდინარე შეგვიძლია ვთქვათ, რომ გარემო და ის ფაქტორები, რომლებიც პირდაპირ კავშირშია გარემოსთან ძალიან დიდ ზეგავლენას ახდენს მთლიან პროცესზე. ეს ყოველივე აუცილებელად არის გასათვალისწინებელი ნებისმიერი სისტემისათვის.

არსებობს ისეთი ტიპის კომპონენტები, რომლებიც ერთიანობაში სრულყოფილებას ანიჭებენ ინტელექტუალურ სისტემას:



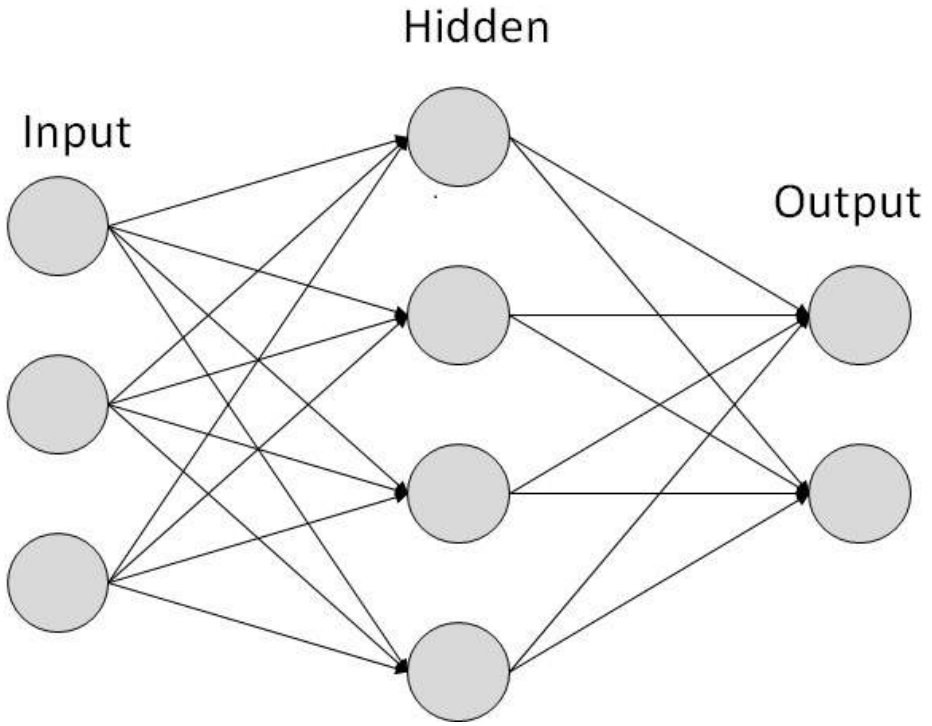
სურ. 19 ინტ.სისტემის სრულყოფილებისათვის სხვადასხვა საჭირო კომპონენტები

ჩვენს ამოცანაში ამ კომპონენტებიდან აუცილებლად დაგვჭირდება:

- **Machine learning** - მანქანური სწავლება, რომლის ალგორითმებსაც გამოვიყენებთ აღნიშნული ამოცანის გადასაჭრელად;
- **Optimization** - წარმოადგენს ისეთი ტიპის კომპონენტს, რომელიც აუცილებელია არა მხოლოდ ინტელექტუალური სისტემის განვითარებისათვის, არამედ უბრალოდ პროგრამებისთვისაც. მისი საშუალებით ხდება მთლიანი სისტემების დახვეწა, წარმადობის გაზრდა რაც საბოლოო ჯამში ეფექტურს ხდის მთლიან სისტემას;
- **Computer vision** [22]- აქ ძირითადად იგულისხმება მანქანის თვალით აღქმული სამყარო. ამ კომპონენტში უმთავრესად მოიაზრება ვიდეო გამოსახულების ანალიზი და მის საფუძველზე წარმოქმნილი გარკვეული ინფორმაციის დაჯგუფება. შესაძლოა ეს არ იყოს მხოლოდ ვიდეო გამოსახულებიდან მიღებული ინფორმაცია, არამედ ის წარმოადგენდეს მანქანისათვის გასაგებ ფორმატში ჩაწერილ რაიმე ინფორმაციას, რომლის აღქმაც უმთავრესაც მანქანას შეუძლია;

Wifi ს საშუალებით პიროვნების იდენტიფიკაციის ამოცანა შეგვიძლია განვაზოგადოთ და წარმოვადგინოთ როგორც გარკვეული შემავალი და გამომავალი ინფორმაცია, რომლებიც გარკვეულად არიან

ერთმანეთთან დაკავშირებული, ეს ყოველივე შეგვიძლია სქემატურად შემდეგნაირად წარმოვადგინოთ:



სურ. 20 სისტემა, როგორც შემავალი და გამომავალი ინფორმაციის ერთობა

სურათზე ნათლად ჩანს კიდევ ერთი , Hidden ტიპის ინფორმაცია (ამ ინფორმაციას მეორენაირად დროებით, Temporary ინფორმაციასაც უწოდებენ). ეს ინფორმაცია შესაძლოა იყოს შედეგი გარკვეული ალგორითმის მუშაობის, რომელიც შემდეგ გამოიყენება სხვა ალგორითმის შემავალ ინფორმაციად, ასევე შესაძლოა იყოს გარკვეულ ბიჯზე მიღებული კონკრეტული შედეგი და ა.შ.

ის წარმოადგენს ერთ კერძო შემთხვევას იმ სხვადასხვა ამოცანებიდან, რომლებშიც შესაძლებელია გამოვიყენოთ ჩვენს მიერ შემუშავებული მოდელი. აგენტი ამ ამოცანაში არის რასაკვირველია ის პროგრამა, რომელიც ყველა იმ ქმედებას შეასრულებს, რაც აუცილებელია ამ ამოცანის გადასაჭრელად. გარემო შეიძლება იყოს ნებისმიერი სახის (როგორც უკვე აღვნიშნეთ, პროგრამული უზრუნველჰყოფის მიზანი ვერ იქნება გარემოსთან სრული ჰარმონია, არამედ ის მეტწილად დამოკიდებულიც კი არის ასეთ გარემოზე). აღნიშნული ამოცანის

გადაწყვეტა მნიშვნელოვანია რადგან ამის საფუძველზე შესაძლოა შეიქმნას ინტელექტუალური სისტემა, რომელიც მასთან ერთად გადაწყვეტს სხვა მნიშვნელოვან ამოცანებს, ექნება დასწავლის უნარი, შეეძლება თვითგანვითარება და ა.შ.

3. მეთოდები და ალგორითმები

წინა თავებში ჩვენ განვიხილეთ ინტელექტუალური სისტემის რაობა, გამოვყავით მისი ძირითადი ნაწილი და მეტ ნაკლებად დავახასიათეთ ისინი. ერთ-ერთი აუცილებელზე აუცილებელი რაც ინტელექტუალურ სისტემებს ახასიათებს არის მასში გამოყენებული ალგორითმები, რომლებიც შეიძლება აღებულ იქნას სხვადასხვა კატეგორიებიდან. ასეთი კატეგორიები უამრავია თუმცა ძირითადად განვიხილავთ ისეთ ალგორითმებს, რომლებიც შესაძლოა შემდეგ გამოვიყენოთ საკუთარი მოდელის აგების დროს.

სანამ უშუალოდ ალგორითმების განხილვაზე გადავალთ, ჯერ განვსაზღვროთ თუ რას წარმოადგენს მანქანური სწავლება. მანქანური სწავლების დანიშნულებაა რაიმე სისტემის (ობიექტის) შემავალი და გამომავალი სიდედეების, მდგომარეობების დაკვირვებათა ნაკრებიდან შედგეს ამ სისტემის ან ობიექტის მოდელი, რომლის საშუალებითაც შესაძლებელი იქნება სხვდასხვა ამოცანების (პროგნოზირება, გადაწყვეტილების მიღება და სხვა) გადაწყვეტა. ინტელექტუალური სისტემები ეფუძვნება მანქანურ სწავლებას (სხვა მნიშვნელოვან ასპექტებთან ერთად) აქედან გამომდინარე უდიდესი მნიშვნელობა აქვს შესაბამისი მოდელის აგებას. ის შეიძლება ეფუძვნებოდეს მხოლოდ კონკრეტული ტიპის ინფორმაციას, ან/და ერგებოდეს მხოლოდ ერთ რომელიმე ინტელექტუალურ სისტემას. განვიხილავთ ზოგად მოდელს, რომელიც წარმოდგენილი იქნება სხვადასხვა ალგორითმების კომბინაციით და განსაზღვრული იქნება სხვადასხვა ტიპის ინტელექტუალური სისტემისათვის.

სიდიდეებს. ნებისმიერი ასეთი სისტემისათვის (როგორიც არ უნდა იყოს ის) ხორციელდება დაკვირვებები (განსხვავებულ გარემო პირობებში, განსხვავებული შემავალი ინფორმაციისათვის) რის შედეგადაც მანქანური სწავლების საფუძველზე ხდება გარკვეული მოდელის შემუშავება. ასეთი

ტიპის მოდელები შეიძლება მორგებულ იქნას ერთ რომელიმე კონკრეტულ სისტემაზე ან იყოს ზოგადი. მანქანური სწავლების პროცესი შედგება შემდეგი ეტაპებისაგან;

1. სასწავლო მონაცემების ნაკრებების ფორმირება;
2. სწავლება სხვადასხვა ალგორითმების გამოყენებით;
3. მიზნობრივი ამოცანის ამოხსნა;

სასწავლო მონაცემების ნაკრებების ფორმირება - გულისხმობს გარკვეული მონაცემების შეგროვებას, რომელსაც შემდეგ გამოვიყენებთ როგორც შემავალი ინფორმაცია. ეს მონაცემთა ნაკრები შესაძლოა აღებული იყოს ნებისმიერი სახით და იყოს ნებისმიერი ტიპის ინფორმაცია.

სწავლება სხვადასხვა ალგორითმების გამოყენებით - არსებობს სხვადასხვა ტიპის ალგორითმები, რომლებიც გამოიყენება მანქანურ სწავლებაში (მაგ: კლასიფიკაციის, ასოციაციის, კლასტერიზაციის და ა.შ) თუმცა მათი პირდაპირი გამოყენება არ იძლევა ეფექტურ შედეგს, აქედან გამომდინარე შეგვიძლია მოვახდინოთ ამ ტიპის ალგორითმებების გამოყენება კომბინირებული სახით რაც უფრო ეფექტურ შედეგს მოგცემს.

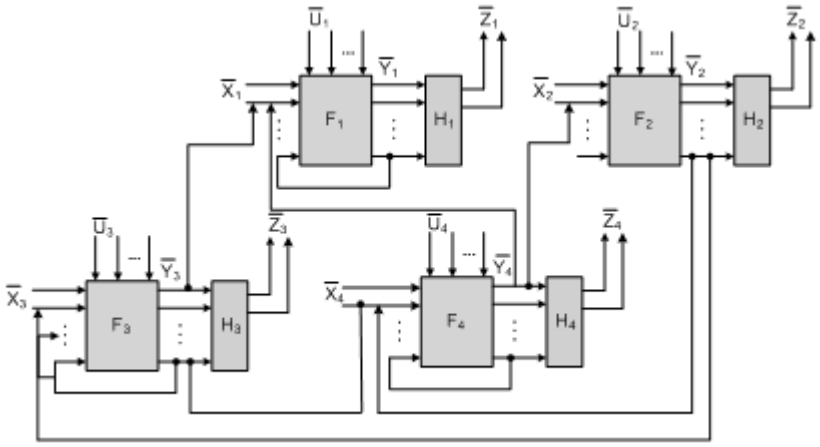
მიზნობრივი ამოცანის ამოხსნა - გულისხმობს მანქანური სწავლების საფუძველზე კონკრეტული ამოცანის გადაწყვეტას, სადაც გამოყენებული იქნება მანქანური სწავლების ალგორითმები კომბინირებული სახით.

განვიხილოთ ორი ძირითადი მოდელი:

- Blackbox ტიპის მოდელი;
- Greybox ტიპის მოდელი;

Blackbox ტიპის მოდელისათვის (როგორი სისტემაც არ უნდა იყოს ის) ცნობილია მხოლოდ შემავალი და გამომავალი სიდიდეები, აქედან გამომდინარე მხოლოდ ვაკვირდებით შემავალი ინფორმაციის საფუძველზე რა გამომავალი ინფორმაციის მიღება ხდება. ასეთი მოდელის სემანტიკა უცნობია, შესაბამისად ვერ მოხერხდება კონკრეტული მეთოდოლოგიის შემუშავება მისი მოდელის გასაუმჯობესებლად.

Greybox ტიპის მოდელისათვის წინასწარაა ცნობილი როგორც შემავალი და გამომავლი ინფორმაცია, ასევე სემანტიკა, რაც გულისხმობს ისეთ საკითხებს, როგორიცაა: სისტემის ობიექტებს შორის კავშირები, როგორ მიმართებაში არიან ეს ობიექტები ერთმანეთთან, როგორ მუშაობს ერთიანობაში ეს სისტემა.



სურ. 21 Greybox მოდელის შესაბამისი სისტემა

სანამ მოდელის აგება/შემუშავება მოხდება, წინასწარ შეგვიძლია განვსაზღვროთ კონკრეტულ შემავალ ინფორმაციაზე როგორი რეაგირება ექნება სისტემას, როგორ გამომავალ ინფორმაციას მოგვცემს ის, რაც ამარტივებს მოდელის აგებას და გვაძლევს შესაძლებლობას ავაგოთ ზუსტი მოდელი მანქანური სწავლებისათვის.

Greybox ტიპის სისტემების შიდა სტრუქტურის ასაგებად შეგვიძლია გამოვიყენოთ სხვადასხვა სწავლების ალგორითმები, ისეთი როგორიცაა K-საშუალო, ასოციაციის, კლასიფიკაციის ალგორითმები, ხოლო შემდეგ მოვახდინოთ მათი კომბინაცია.

სანამ უშუალოდ კომბინაციაზე გადავალთ, განვიხილოთ რამოდენიმე საინტერესო ალგორითმი, რომელიც შემდეგ შეიძლება გამოვიყენოთ კომბინირებული სახით.

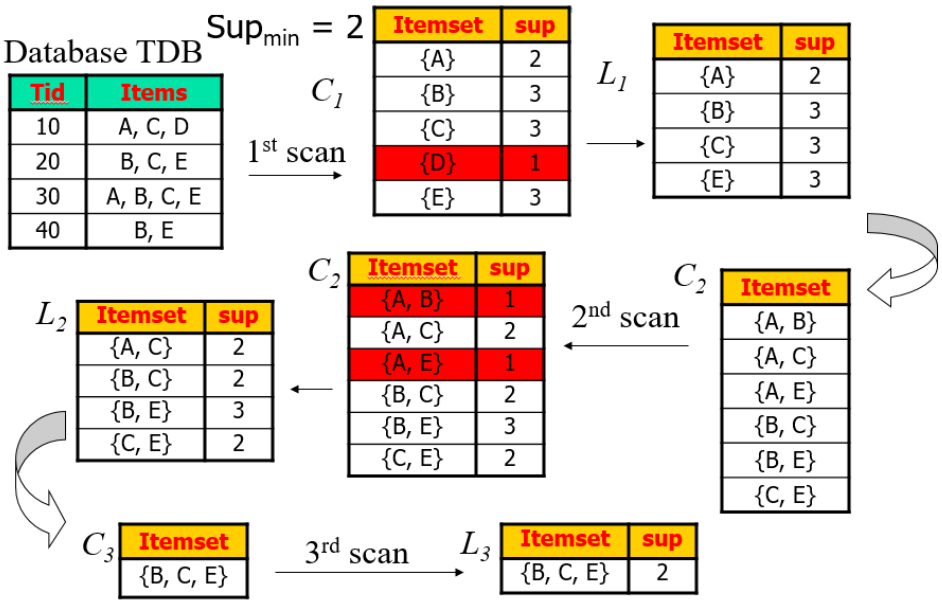
სანამ უშუალოდ კომბინაციაზე გადავალთ, განვიხილოთ რამოდენიმე საინტერესო ალგორითმი, რომელიც შემდეგ შეიძლება გამოვიყენოთ კომბინირებული სახით.

ასოციაციის პრინციპი არის ზოგადი წესი, რომელიც მოიცავს კონკრეტულ ალგორითმებს. ერთ ერთს წარმოადგენს აპრიორის ალგორითმი [7]. ის მდგომარეობს შემდეგში: მოცემულია გარკვეული მონაცემთა ბაზა, რომელსაც პირობითად ეწოდება TDB (Transaction Database). ამ ბაზაში ასახულია გარკვეული ტრანზაქციები/ჩანაწერები და თითოეულ მათგანში გაერთიანებულია სხვადასხვა მონაცემის კომბინაცია.

- Sup_{min}- კოეფიციენტი;
- C_k– k რაოდენობის შესაძლო მონაცემი;
- L_k – k რაოდენობის სიხშირული მონაცემი;

C_k სიმრავლეში ერთიანდება ყველა ის მონაცემი ინდივიდუალურად, რომლებიც ერთმანეთთან კომბინაციაში იყვნენ საწყის მონაცემთა ბაზაში. მაგ: საწყის ბაზაში არის სამი ჩანაწერი: {A, C, D}, { B, C, E}, {A, B, C, E}, {B, E} C_kსიმრავლეში იქნება აღებული ინდივიდუალური მონაცემები ცალ-ცალკე {A}, {B}, {C}, {D}, {E} ხოლო თითოეულისათვის გვექნება Sup კოეფიციენტი, რომელიც განისაზღვრება იმ რიცხვით რა რაოდენობითაც გვხვდება ეს მონაცემი საწყისი ბაზის ჩანაწერებში. მაგ: ზემოაღნიშნულ მაგალითში {A} გვხვდება 2 ჯერ, B – 3 ჯერ, C – 3 ჯერ, D – 1 ხელ, ხოლო E – 3 ჯერ, ამიტომშესაბამისი Sup კოეფიციენტები იქნება: SUP(A) = 2, SUP(B) = 3, SUP(C) = 3, SUP(D) = 1, SUP(E) = 3. **აღნიშნული სიმრავლიდან ვიღებთ L_k ქვესიმრავლეს, სადაც ერთიანდება ყველა ის მონაცემი რაც იყო C_k სიმრავლეში და ამავდროულად თითოეულის Supკოეფიციენტი მეტია ან ტოლია მოცემულ Sup_{min}ზე**. შემდეგ ეტაპზე ვიღებთ მიღებული L_k სიმრავლიდან ახალ C_k სიმრავლეს სადაც ვახდენთ თითოეული ჩანაწერის/მონაცემის ინდივიდუალურ გაერთიანებას სხვა დანარჩენ მონაცემთან და ამ სიმრავლისათვის ვპოულობთ ახალ SUP კოეფიციენტებს. მაგ: თუ გვაქვს {A, C} სიმრავლე და ეს სიმრავლე საწყისი ბაზის ჩანაწერებში გვხვდება 2 ჯერ, როგორც 2 რომელიმე ჩანაწერის ქვესიმრავლე, ასეთ შემთხვევაში მისი Sup კოეფიციენტი იქნება 2. აღნიშნული პროცესი

გრძელდება მანამ, სანამ არ დავალთ ისეთ სიმრავლეზე რომლიდანაც ახალი სიმრავლის მიღება აღარ არის შესაძლებელი.



სურ. 22 აპრიორის ალგორითმი, კონკრეტული მაგალითი

შეგვიძლია მოვიყვანოთ აღნიშნული მეთოდის ფსევდოკოდიც:

C_k: kრაოდენობის შესაძლო მონაცემები;

L_k: kრაოდენობის სიხშირული მონაცემები;

L_1= {სიხშირული მონაცემები};

for (k=1; L_k!=∅; k++) **do begin**

C_{k+1}=L_k დან მიღებული შესაძლო მონაცემები;

foreach თითოეული t ტრანზაქციისთვის გაიზარდოს რაოდენობა C_{k+1}ის ყველა მონაცემის, რომლებიც ეკუთვნიან t სიმრავლეს

L_{k+1}= შესაძლო მონაცემები C_{k+1}ში, რომელთაც გააჩნიათ min_support

end

return $\cup_k L_k$;

ამ პრინციპის გამოყენებით შეგვიძლია Greybox ტიპის სისტემისთვის გამოვთვალოთ sup მნიშვნელობა, რომელიც შემდეგ დაგვეხმარება მოდელის შედგენაში.

კლასიფიკაციის ერთ-ერთ კლასიკურ ალგორითმს წარმოადგენს გადაწყვეტილებათა ხეები (Decision Tree)[8]. საწყის ეტაპზე მოცემულია:

- მატრიცა (რომელიც შედგება გარკვეული ვექტორ-სვეტებისაგან);
- ასევე საწყისი ენტროპია (გამოთვლილი სახით);

ალგორითმი მუშაობს შემდეგი პრინციპით:

- თითოეული ვექტორ-სვეტისთვის გამოვთვალოთ საშუალო ენტროპია AE_i;
- გამოთვლილი საშუალო ენტროპიებიდან ამოვარჩიოთ მინიმალური რის შედეგადაც მოხდება წინა ეტაპზე მიღებული მატრიცის გაყოფა მარცხენა და მარჯვენა ქვე მატრიცებად, ხოლო არჩეული ვექტორ-სვეტი კი იქნება ხის მიმდინარე კვანძი;
- გავიმეოროთ ეს პროცესი მანამ სანამ არ მოხდება მთელი ხის აგება;

საშუალო ენტროპიების დათვლა თითოეული ვექტორ სვეტისთვის ხდება შემდეგნაირად:

1. f_i ვექტორ-სვეტისთვის დავთვალოთ დადებითი და უარყოფითი ელემენტების რაოდენობა მარცხენა და მარჯვენა კვანძისთვის (თითოეულიf_i წარმოადგენს ხის კვანძს, რომელსაც გააჩნია მარცხენა(დადებითი კვანძი) და მარჯვენა (უარყოფითი კვანძი)). **მარცხენა კვანძისთვის დადებითი ელემენტების რაოდენობა** ტოლია იმ 0 ის ტოლი მნიშვნელობების რაოდენობის, რომლის შესაბამისი y არის 0. **მარცხენა კვანძისათის უარყოფითი ელემენტების რაოდენობა** ტოლია იმ 0 ის ტოლი მნიშვნელობების რაოდენობის, რომლის შესაბამისი y არის 1. **მარჯვენა კვანძისთვის დადებითი ელემენტების რაოდენობა** ტოლია იმ 1 ის ტოლი მნიშვნელობების რაოდენობის, რომლის შესაბამისი y არის 0. **მარჯვენა კვანძისათის უარყოფითი ელემენტების რაოდენობა**

ტოლია იმ 1 ის ტოლი მნიშვნელობების რაოდენობის, რომლის შესაბამისი y არის 1.

2. რადგან ორობით სისტემაზეა საუბარი უნდა გამოვთვალოთ შემდეგი ენტროპიები: $I_i(p_0,n_0) = - \frac{p_0}{p_{0+}n_0} log_2(\frac{p_0}{p_{0+}n_0}) - (1 - \frac{p_0}{p_{0+}n_0}) log_2(1 - \frac{p_0}{p_{0+}n_0})$ და $I_i(p_1,n_1) = - \frac{p_1}{p_{1+}n_1} log_2(\frac{p_1}{p_{1+}n_1}) - (1 - \frac{p_1}{p_{1+}n_1}) log_2(1 - \frac{p_1}{p_{1+}n_1})$;
3. გამოვთვალოთ AE_i რომელიც განისაზღვრება შემდეგნაირად: $AE_i = \frac{p_0+n_0}{p+n} I_i(P_0,n_0) + \frac{p_1+n_1}{p+n} I_i(P_1,n_1)$;

კონკრეტულ მაგალითზე შეგვიძლია მოვახდინოთ სიმულაცია და ვნახოთ თუ როგორ აიგება გადაწყვეტილებათ ხე, დაუშვათ მოცემულია საწყისის მატრიცა:

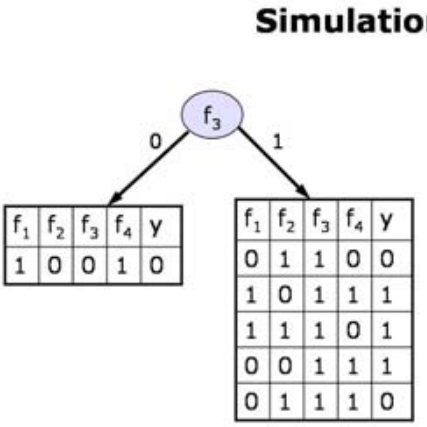
Simulation

- H(D) = .92
- AE_1=.92, AE_2=.92, AE_3=.81, AE_4=1

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
1	0	0	1	0
0	1	1	1	0

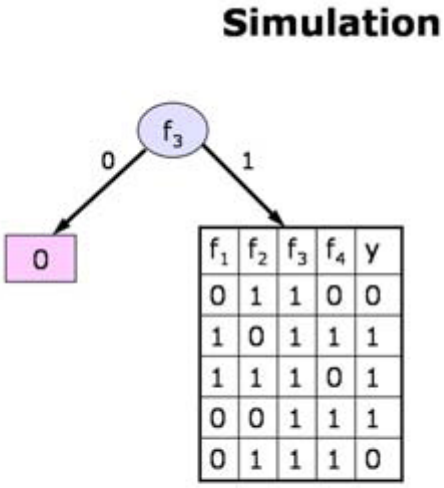
სურ. 23 გადაწყვეტილებათა ხეები, საწყისი მატრიცა

მოცემული მაგალითიდან ჩანს, რომ საწყისი ენტროპია არის 0.92. ვითვლით სათითაო ენტროპიებს თითოეული ვექტორ სვეტისას და ვირჩევთ მათგან მინიმალურს. მოცემულ შემთხვევაში საუკეთო არის f_3, შესაბამისად ვირჩევთ ამ კვანძს და ვახდენთ ამ კვანძე მატრიცის გაყოფას.



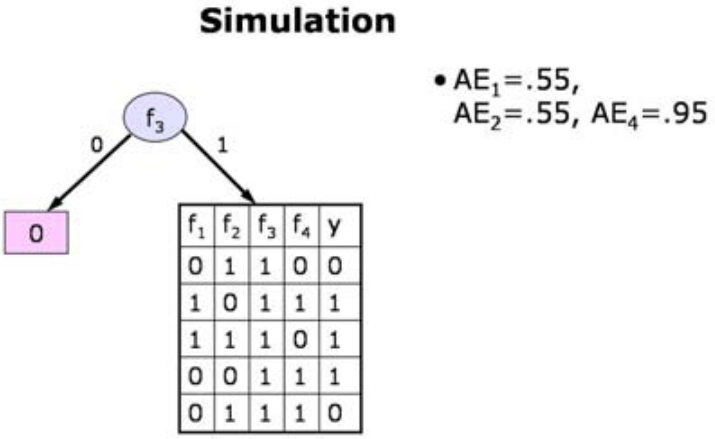
სურ. 24 ავირჩიეთ კვანძი და მოვახდინეთ ორ სიმრავლედ გაყოფა საწყისი მატრიცის

მარცხენა მხარეს მიღებული მატრიცა არის ერთ სტრიქონიანი, რომლის მნიშვნელობაც არის 0, აქედან გამომდინარე ის გადაიქცევა „ფოთლად“ , რომლის მნიშვნელობაც იქნება 0.



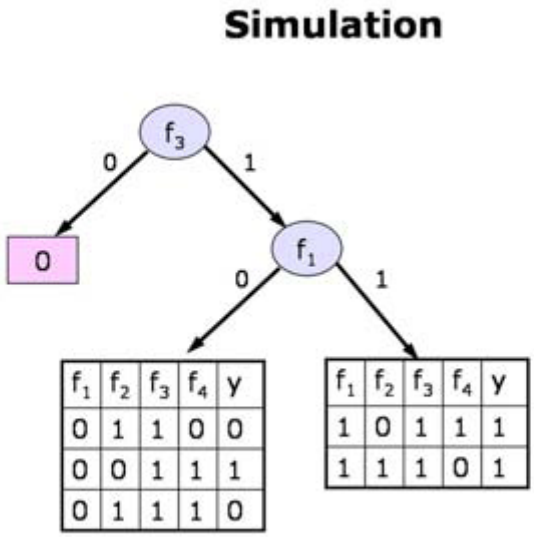
სურ. 25 მარცხენა მატრიცადან მიღებული მნიშვნელობა

იგივე პროცესი გაგრძელდება მარჯვენა მატრიცისთვის, დავითვლით საშუალო ენტროპიებს და მოვახდენთ მის გაყოფას:



სურ. 26 პროცესის გაგრძელება მარცხენა მატრიცისთვის

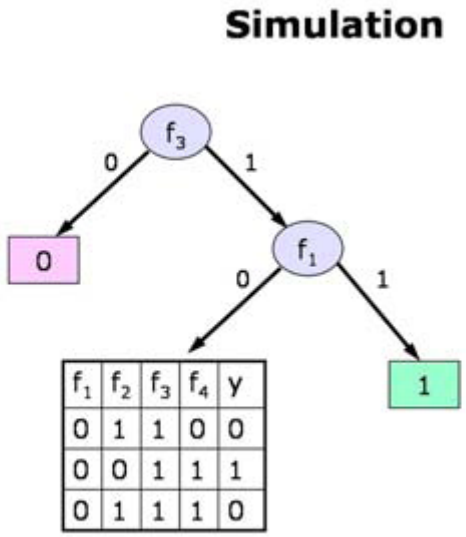
ამ კონკრეტულ შემთხვევაში პირველ და მეოთხე ვექტორ სვეტებს გააჩნიათ ზუსტად ერთი და იგივე მნიშვნელობა, შესაბამისად არ აქვს მნიშვნელობა, რომელ მათგანს ავირჩევთ.



სურ. 27 მატრიცის კიდევ ერთხელ გაყოფა

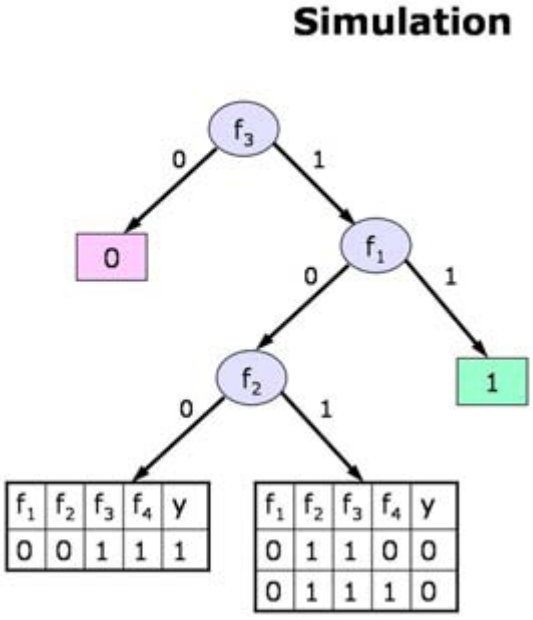
რადგან პირველი და მესამე ვექტორ სვეტის შესაბამისი საშუალო ენტროპია იყო ერთი და იგივე (მნიშვნელობა არ აქვს ასეთ დროს რომელს ავირჩევთ), ავირჩიეთ პირველი და მოვახდინეთ მისი გაყოფა. ამის შემდეგ კი პროცესი

გრძელდება რეკურსიულად ყველა დარჩენილი მატრიცისთვის მანამ სანამ არ აიგება მთელი ხე.



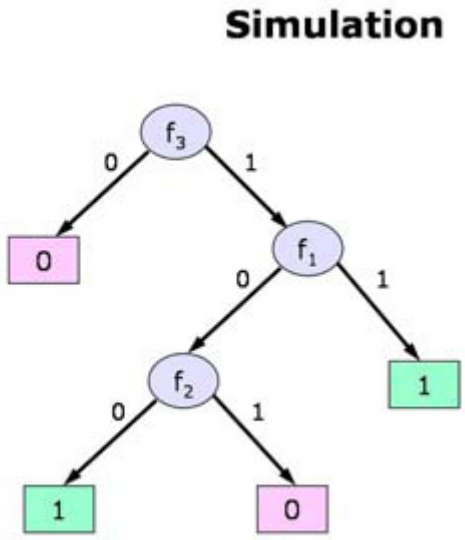
სურ. 28 პროცესის გაგრძელება დარჩენილი მატრიცისთვის

წინა სურათში მიღებული მატრიცაც კვლავ გაიყოფა



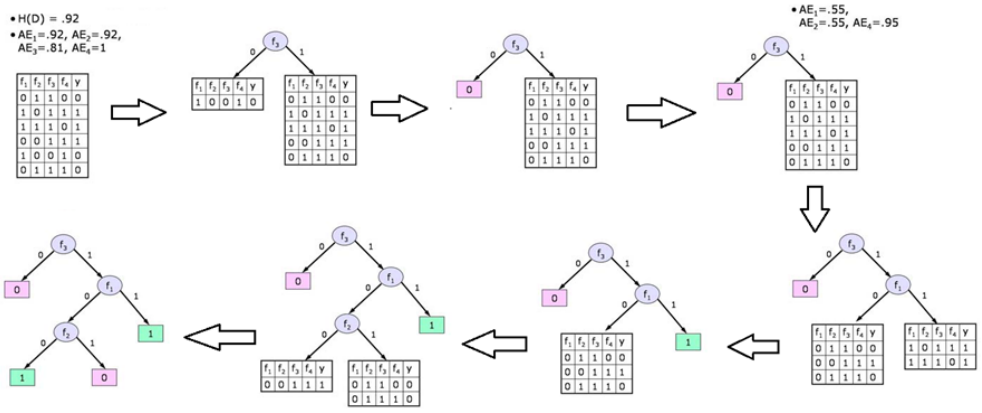
სურ. 29 წინა ეტაპზე მიღებული მატრიცა კვლავ გაიყო

საბოლოო ჯამში კი მივიღეთ შემდეგი აგებული ხე:



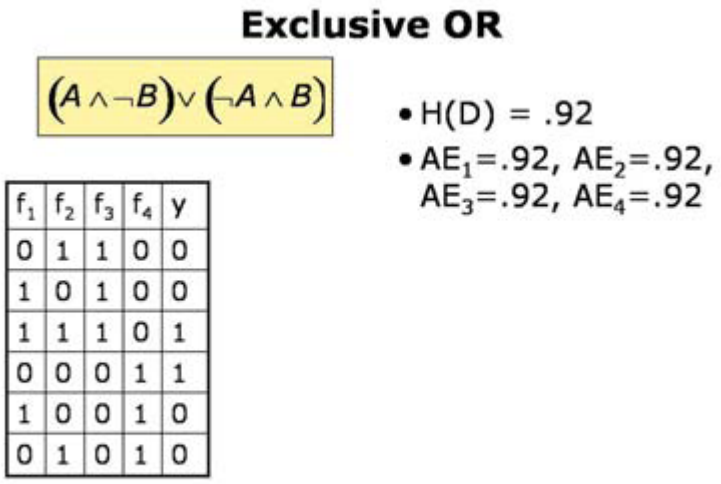
სურ. 30 საბოლოოდ აგებული ხე

მთლიანობაში პროცესი შეგვიძლია შემდეგნაირად წარმოვადგინოთ:



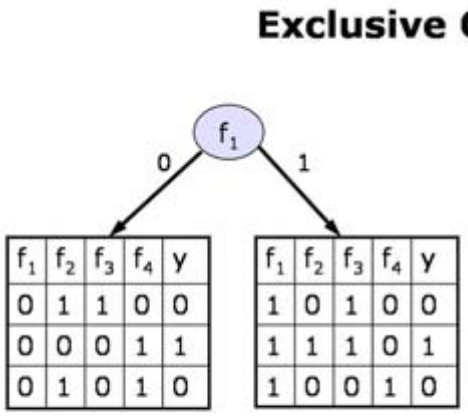
სურ. 31 გადაწყვეტილებათა ხის აგება, მთლიანი პროცესი

ასეთი ტიპის გადაწყვეტილებათ ხეებში შეგვიძლია გამოვიყენოთ ე.წ XOR ოპერაცია ხის ასაგებად. მისი ფორმულა შემდეგში მდგომარეობს. მოცემული ორი პრედიკატიდან ამ ოპერაციის შედეგი უდრის 0 ს მაშინ და მხოლოდ მაშინ როდესაც პირველი და მეორე პრედიკატები არის 0 (ორივე ერთად) ან არის 1 (ორივე ერთად), წინააღმდეგ შემთხვევაში ოპერაციის შედეგი არის 1.



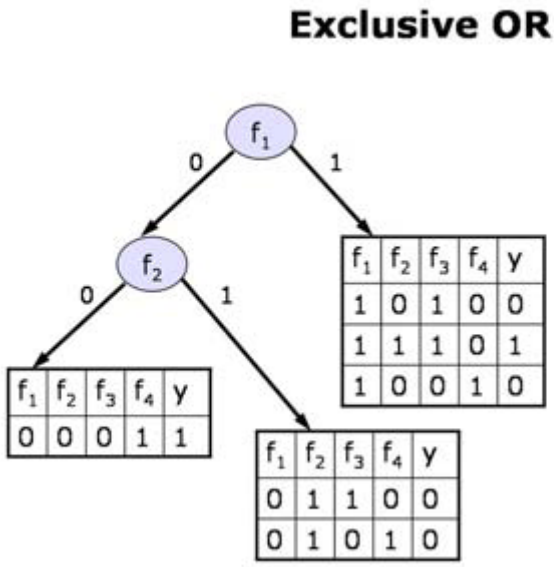
სურ. 32 xor ოპერაციით ხის აგება

რადგან ენტროპიები ტოლია ავიღოთ პირველი ვექტორი და მოვახდინოთ გაყოფა მასზე (მატრიცის გახლეჩა).



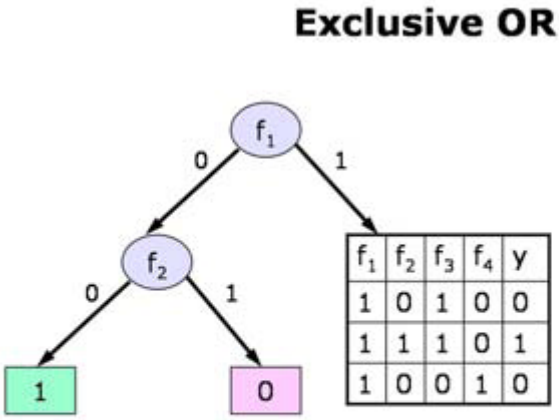
სურ. 33 მატრიცის საწყისი გაყოფა

შემდეგ მიღებული მატრიცები გავყოთ ისევ რეკურსიულად, რომლის შედეგადაც მივიღებთ:



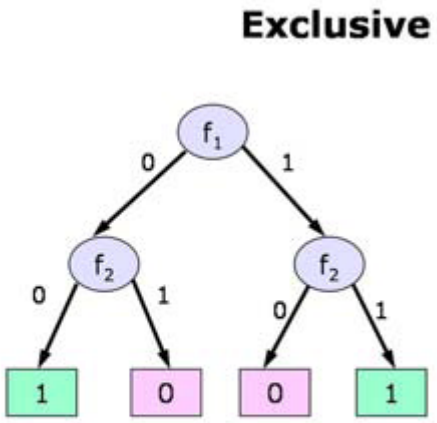
სურ. 34 მატრიცების გაყოფა (გაგრძელება)

შემდეგ ისევ ვახდენთ მატრიცების გაყოფას მანამ სანამ არ მივიღებთ სრულ ხეს.



სურ. 35 მატრიცის გაყოფა (გაგრძელება)

საბოლოო ჯამში მიღებული ხე შემდეგნაირად გამოიყურება:



სურ. 36 xor ოპერაციის გამოყენებით მიღებული გადაწყვეტილებათა ხე

აღნიშნული მეთოდი საშუალებას გვაძლევს გარკვეული შემავალი ინფორმაციის საშუალებით ავაგოთ გადაწყვეტილებათა ხე რაც საბოლოო ჯამში გვეხმარება მოდელის სტრუქტურის აგებაში.

კლასტერიზაცია არის მონაცემთა წერტილების ისეთ სიმრავლეებად დაყოფა სადაც:

- წერტილები თითოეულ დაყოფილ ჯგუფში გარკვეული თვისების მიხედვით ძალიანახლოს გვანან ერთმანეთთან;
- წერტილები სხვადასხვა ჯგუფებში არიან მაქსიმალურად განსხვავებულები (გვაქვს გარკვეულ თვისების მქონე მონაცემთა ჯგუფები);

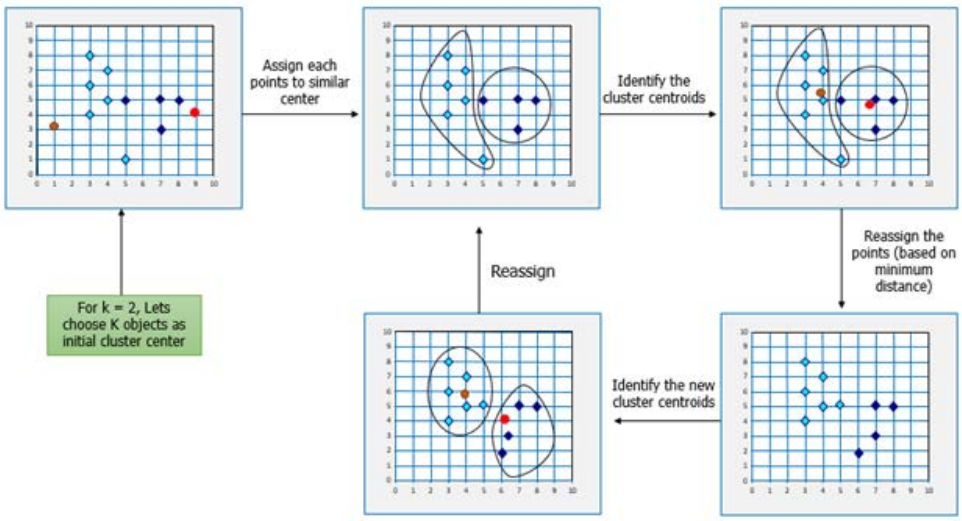
მოცემულ წერტილთა სიმრავლისათვის უნდა მოხდეს დაჯგუფება მათი მსგავსების მიხედვით (უნდა აიგოს ისეთი კლასტერები სადაც თითოეულ ჯგუფში რაღაც თვისების მიხედვით იქნება მაქსიმალურად ერთნაირი ობიექტები). ძირითადი სქემა კლასტერიზაციის შეგვიძლია დავყოთ შემდეგ ძირითად კომპონენტებად:

სურ. 37 კლასტერიზაციის ზოგადი სქემა

K-means ალგორითმი წარმოადგენს კლასტერიზაციის პრინციპის ერთ-ერთ ალგორითმს.მოცემული გვაქვს k და წერტილების სიმრავლე, რომელთაგანაც უნდა აიგოს კლასტერები.ალგორითმი კი წარმოადგენს შემდეგი ბიჯების ერთობლიობას:

- დავყოთ ობიექტები k რაოდენობის არა ცარიელ ქვესიმრავლეებად;
- ვიპოვოთ დაყოფილი ქვესიმრავლეების/კლასტერების ცენტროიდები;
- თითოეული წერტილი/ობიექტი მივაკუთნოთ კონკრეტულ კლასტერს;
- გამოვთვალოთ მანძილები თითოეული წერტილიდან და ამ კლასტერისთვის გამოყოფილი სხვა მრავალი წერტილიდან კლასტერამდე სადაც მანძილი ცენტროიდამდე არის მინიმალური;

წერტილების გადანაწილების შემდეგ (წინა პუნქტის შედეგად ხდება გარკვეული წერტილების სხვა კლასტერზე მინიჭება, რადგან მის ცენტროიდთან უფრო ახლოს არის ეს წერტილი) ვიპოვოთ ცენტროიდი ახალ კლასტერში [9].



სურ. 38 K-Means ალგორითმის სიმულაცია

მათემატიკური მოდელის სახით კი გვექნება:

1. მოვახდინოთ კლასტერის ცენტროიდების ინიციალიზაცია
2. ვიმეორებთ ქვემოთ მოყვანილ ბიჯებს მანამ სანამ არ მივიღებთ ოპტიმალურ განაწილებას:
   a. ყოველი i სთვის, ვპოულობთ $C^{(i)}$:=arg min_j $||x^{(i)} - \mu_j||^2$
   b. ყოველი j სთვის კი $\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\}x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}}$

ჩვენ განვიხილეთ რამოდენიმე ტიპის სხვადასხვა ალგორითმი, მოვიყვანეთ მცირე მაგალითები. როგორი ტიპის ალგორითმებსაც არ უნდა ეხებოდეს საქმე და როგორც არ უნდა მოვახდინოთ მათი ოპტიმიზაცია ინფორმაცია შეიძლება დაიყოს ორ კატეგორიად:

1. **შემავალი ინფორმაცია** – ნებისმიერი სახის ინფორმაცია, რომელიც ადექვატურად მიეწოდება შესაბამის ალგორითმს;
2. **გამომავალი ინფორმაცია** – ნებისმიერი სახის ინფორმაცია. აუცილებელია აღნიშნული ინფორმაცია ადექვატურად იქნას წარმოდგენილი, რადგან ის შეიძლება გამოყენებულ იქნას როგორც სხვა ალგორითმის შემავალი მონაცემები ან/და საბოლოო შედეგი, რომელზე დაყრდნობითაც ადამიანმა უნდა მიიღოს გარკვეული გადაწყვეტილება.

საწყის ეტაპზე ხდება ინფორმაციის კატეგორიის იდენტიფიკაცია (ინფორმაცია შეიძლება იყოს მრავალი სახის, აქედან გამომდინარე პირველ რიგში აუცილებელია იმის გარკვევა თუ რომელ კატეგორიას ეკუთვნის ის). შემავალი იქნება ინფორმაცია თუ გამომავალი, აუცილებელია ის გარდაიქმნას კონკრეტულ ფორმატში. აღნიშნული ფორმატს გამოიყენებს ალგორითმი ინფორმაციის იდენტიფიკაციისთვის. მონაცემთა წარმოდგენა ისე უნდა განხორციელდეს, რომ საბოლოო შედეგის აღქმა შეეძლოს არა მხოლოდ ინტელექტუალურ სისტემას, არამედ ადამიანსაც, რომელიც იყენებს აღნიშნულ სისტემას გარკვეული ამოცანების შესასრულებლად. კონკრეტულ ფორმატში გარდაქმნა სხვა არაფერია თუ არა მონაცემების ნორმალიზაცია/დენორმალიზაცია, რომელსაც განვიხილავთ. ცალკე საკითხია რამდენად კარგია შერჩეული ნორმალიზაცია/დენორმალიზაციის ალგორითმები თუმცა ზოგადად ნორმალიზაცია/დენორმალიზაციის გარეშე ვერ მოვახდენთ მთლიანი სისტემის მუშაობის ადექვატურობას.

არსებობს მონაცემთა სხვადასხვა ტიპები, რომლებიც გამოიყენება ნებისმიერი ალგორითმისთვის [1]. შემავალი და გამომავალი ინფორმაცია მიეკუთვნება ამ ტიპებიდან ერთ-ერთს. ძირითადად განსაზღვულია შემდეგი მონაცემთა ტიპები:

ნომინალური – მოიცავს ისეთი ტიპის ინფორმაციას, რომლის ყველა შესაძლო მნიშვნელობა წინასწარ არის ცნობილი, შესაბამისად მიმდინარე მნიშვნელობა აუცილებლად იქნება ამ ჩამოთვლილი მნიშვნელობებიდან ერთ-ერთი. აღნიშნულის კლასიკური მაგალითია სქესი, რომელსაც გააჩნია ორი შესაძლო მნიშვნელობა: მამრობითი ან მდედრობითი.

ორდინალური – მოიცავს ისეთი ტიპის ინფორმაციას, რომლის ყველა შესაძლო მნიშვნელობა წინასწარ არის ცნობილი, მიმდინარე მნიშვნელობა კი ხასიათდება ცვალებადობით რაღაც პირობებიდან გამომდინარე. აღნიშნულის კლასიკური მაგალითია ტემპერატურის სამი შესაძლო კატეგორია: “ცხელი”, “თბილი”, “ცივი”. პირველ ჯერზე მიმდინარე მნიშვნელობა შეიძლება იყოს “ცხელი”, გარკვეული პერიოდის/პირობების

შედეგად კი მისი მნიშვნელობა შეიძლება შეიცვალოს და გახდეს “თბილი” და ა.შ. აქედან გამომდინარე ძირითადი განსხვავება ნომინალურსა და ორდინალურ ინფორმაციას შორის არის მიმდინარე მნიშვნელობის ცვალებადობა, (რაიმე პირობის/მდგომარეობის გათვალისწინებით) რომელიც ახასიათებს ორდინალური ტიპის ინფორმაციას.

ინტერვალური – მოიცავს რიცხვითი ტიპის ინფორმაციას, რომელსაც გააჩნია საწყისი 0. აღნიშნულის მაგალითია რაიმე წელიწადი, წარმოდგენილი რიცხვით ფორმატში (მაგალითად, 2010 წელი და ა.შ).

ფარდობითი – მოიცავს რიცხვითი ტიპის ინფორმაციას, რომელსაც არ გააჩნია 0 ზე დაბალი ან ტოლი მნიშვნელობა. აღნიშნულის კლასიკური მაგალითია ასაკი (მაგალიტად, 10 წელი, არ არსებობს 0 წელი ან თუნდაც -1 წელი).

როგორიც არ უნდა იყოს მონაცემები, მათ ახასიათებთ გარკვეული ოპერაციების მხარდაჭერა. მაგ: შეკრება, გამოკლება, გამრავლება და გაყოფა. ასევე შეიძლება ახასიათებდეთ მიმართებითი დამოკიდებულებებიც, მაგ: მეტია, ნაკლებია, ტოლია და ა.შ.

	ნომინალური	ორდინალური	ინტერვალური	ფარდობითი
*აწ /	არა	არა	არა	დიახ
+აწ -	არა	არა	დიახ	დიახ
<აწ>	არა	დიახ	დიახ	დიახ
=აწ ! =	დიახ	დიახ	დიახ	დიახ
მაგალითი	გეწდერი	ცხელი/თბილი/ცივი	წელიწადი	ასაკი

სურ. 39 მონაცემთა ტიპები

მოცემულ ცხრილში წარმოდგენილია ოთხი სხვადასხვა მონაცემთა ტიპი თუმცა საბოლოო ჯამში შეიძლება განვსაზღვროთ ორი ძირითადი ტიპი, რომელთაგანაც პირველია რაოდენობრივი, ხოლო მეორე ხარისხობრივი. რაოდენობრივს განეკუთვნება ყველა ისეთი ტიპის ინფორმაცია, რომელიც რიცხვითი სახით წარმოდგება, ხოლო

ხარისხობრივს კი სხვა დანარჩენი. ინფორმაციის ნორმალიზაცია/დენორმალიზაცია გულისხმობს რაიმე კონკრეტული ალგორითმით მიმდინარე ინფორმაციის გარდაქმნას. არსებობს უამრავი სხვადასხვა ალგორითმი, რომელთაგანაც ერთ-ერთია Equilateral Encoding.

აღნიშნული ალგორითმისათვის გენერირებული ინფორმაცია წარმოდგენილია მატრიცის სახით, რომლის განზომილებებია NxN-1 N არის კატეგორიების რაოდენობა (N შეიძლება იყოს 1 ან მეტი, აღნიშნულის განხილვისთვის ვიგულისხმოთ რომ N>1), ხოლო თითოეული სტრიქონი შეიცავს ე.წ encoding ს ს მიმდინარე კატეგორიისათვის. ალგორითმისათვის აუცილებელია რაიმე წინასწარგანსაზღვრული შუალედი, რომელშიც მოვახდენთ მონაცემთა გრადაციას (როგორც წესი ეს შუალედი არის -1 დან 1 მდე თუმცა შეიძლება აღებულ იქნას რაიმე განსხვავებულიც. აღნიშნულის განხილვისთვის გამოვიყენოთ შუალედი -1 დან 1 მდე). ალგორითმი შეიძლება წარმოვადგინოთ შემდეგი ბიჯების სახით:

1. საწყისი მატრიცის პირველი სტრიქონის პირველ ელემენტში უნდა ჩაიწეროს შუალედის მინიმალური მნიშვნელობა, ხოლო მეორე სტრიქონის პირველ ელემენტში კი შუალედის მაქსიმალური მნიშვნელობა. შესაბამისად გვექნება:

$$result[\![0]\!][\![0]\!] = -1;$$

$$result[\![1]\!][\![0]\!] = 1;$$

2. ვახდენთ იტერაციას მე 2 კატეგორიიდან ზემოთ N მდე. (უგულებელვჰყობთ პირველს, რადგან არ გვაქვს განხილული ისეთი შემთხვევა როდესაც მოცემულია მხოლოდ 1 კატეგორია)

$$for\ k\ from\ 2\ to\ N\ \{$$

3. ვითვლით გრადაციის კოეფიციენტს. ყოველი ახლად აგებული მატრიცის მიღება ხდება წინა მატრიცის გამოყენებით (ათვლა მეორედან დავიწყეთ, მესამე კატეგორიისათვის მატრიცის მიღება შეგვიძლია წინა მატრიციდან, რომელიც უკვე გენერირებულია)

$$f = \frac{sqrt(N * N - 1)}{r};$$

4. ვახდენთ მატრიცის იმ ნაწილზე იტერაციას, რომელიც უკვე გამოვთვალეთ და მოვახდინეთ მისი გრადაცია

$$for\ i\ from\ 0\ to\ k\ \{$$
$$for\ j\ from\ 0\ to\ k-1\ \{$$
$$result[\![i]\!][\![j]\!] *= f$$
$$\}$$
$$\}$$

5. გამოვთვალოთ მატრიცის ზღვარი (ვექტორ-სვეტების გარკვეული მნიშვნელობები)

$r = -1/N;$

$for\ j\ from\ 0\ to\ k\ \{$

$result[\![i]\!][\![k-1]\!] = r$

$\}$

6. მატრიცის ბოლო მნიშვნელობაში ჩავწეროთ 1 და გავაგრძელოთ მე 2 ბიჯზე აღწერილი ციკლი:

$$result[\![k]\!][\![k-1]\!] = 1;$$

7. როდესაც ციკლი მთლიანად დასრულდება , უნდა მოვახდინოთ მთლიანი მატრიცის გრადაცია [-1 , 1] შუალედში:

$$dataLow = -1;$$
$$dataHigh = 1;$$

$for\ row\ from\ 0\ to\ N\ \{$

$for\ col\ from\ 0\ to\ N-1\ \{$

$result[row][col] = \left(\frac{result\ [row\ ][col\ ]=dataLow}{dataHig\ h-dataLow}\right) * (normalizedHigh - normalizedLow) + normalizedLow;$

}
}

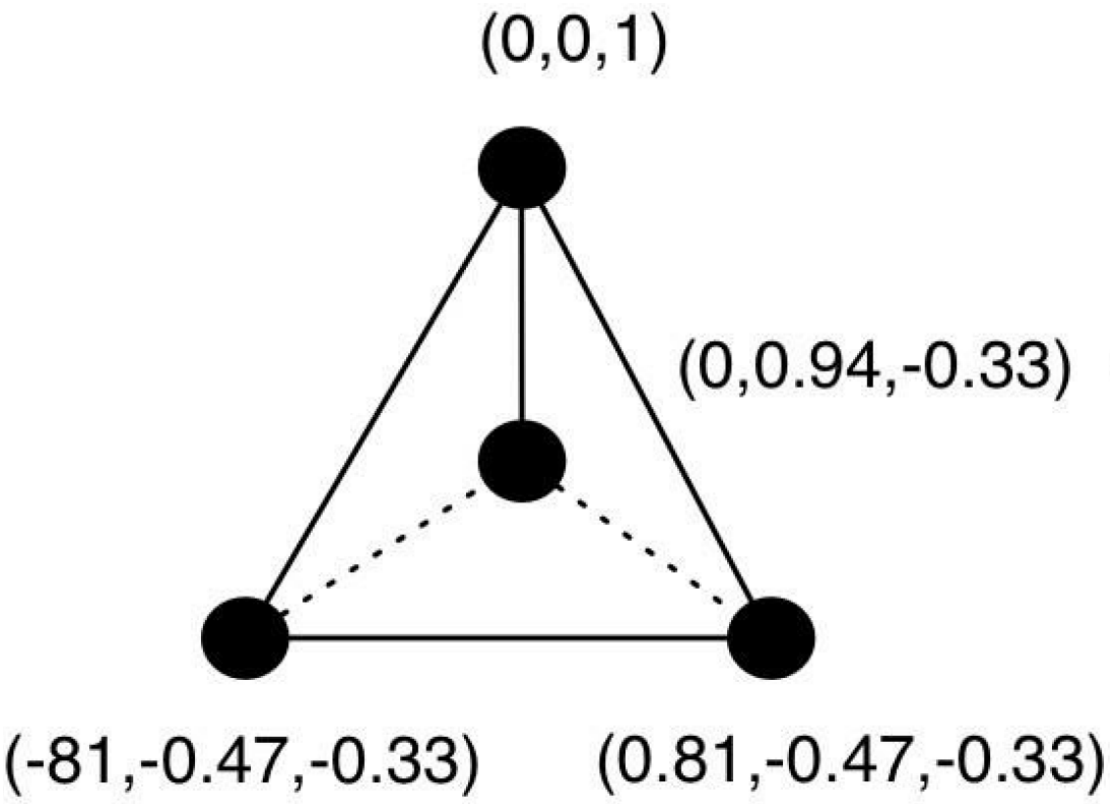
კონკრეტული კატეგორიის სანახავად (რომელიც უკვე ნორმალიზებულია), უნდა ამოვიღოთ შესაბამისი სტრიქონი მიღებული/გენერირებული მატრიციდან. დეკოდირებისთვის (დენორმალიზა-ციისათვის) კი უნდა ვიპოვოთ მატრიცაში ისეთი სტრიქონი, რომელსაც გააჩნია უმცირესი ევკლიდური მანძილი გამომავალ ვექტორთან (ნორმალიზებულ ვექტორთან) მიმართებაში.

ზემოთ აღწერილი ალგორითმის საფუძველზე შეგვიძლია მოვიყვანოთ კონკრეტული მაგალითი: პირობითად ავიღოთ შემდეგი პარამეტრები: კატეგორიების/კლასების რაოდენობა - 5, შუალედი - [-1,1]. ალგორითმის საფუძველზე მიღებული მატრიცა შემდეგნაირად გამოიყურება:

Class	Output 0	Output 1	Output 2	Output 3
Class #1	-0.7905694150420949	-0.4564354645876385	-0.32274861218395134	-0.25
Class #2	0.790569415042095	-0.4564354645876385	-0.32274861218395134	-0.25
Class #3	0	0.912870929175277	-0.32274861218395134	-0.25
Class #4	0	0	0.9682458365518543	-0.25
Class #5	0	0	0	1

სურ. 40 Equilateral Encoding ის მაგალითი

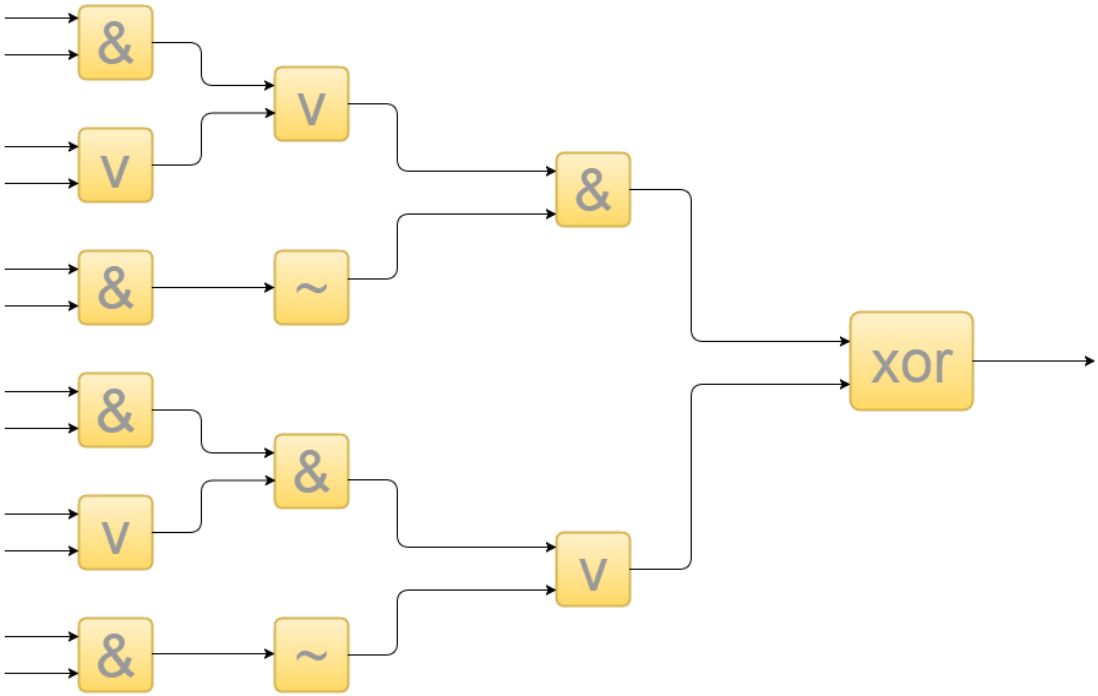
შეგვიძლია მოვიყვანოთ მისი ვიზუალური წარმოდგენა, როდესაც კატეგორიების რაოდენობა არის 4:



სურ. 41 Equilateral Encoding ის ვიზუალური წარმოდგენა

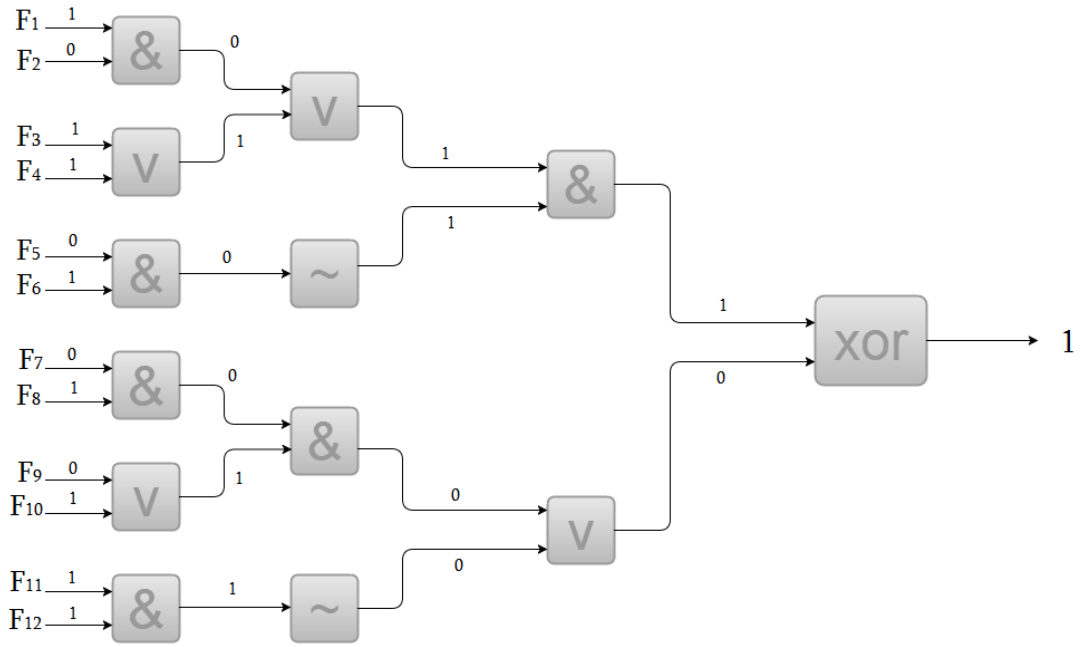
სისტემაში ინფორმაცია დავყავით ორ ძირითად კატეგორიად, რომელთაგანაც ერთ ერთს წარმოადგენს შემავალი ინფორმაცია. სანამ უშუალოდ სისტემა დაიწყებს თავისი ალგორითმების გამოყენებას, აუცილებელია მოხდეს გარდაქმნა მონაცემების (ნორმალიზაცია). ნებისმიერი ალგორითმისათვის აუცილებელია შემავალი ინფორმაცია. როგორც უკვე აღვნიშნეთ გადაწყვეტილებათა ხეების[8] ალგორითმი იყენებს მატრიცას, რომელიც შედგება კონკრეტული რაოდენობის ვექტორ-სვეტისაგან და y ვექტორ-სვეტისაგან. ასევე წინასწარაა მოცემული დათვლილი საშუალო ენტროპიები. შემავალი ინფორმაციიდან ჩვენ შეიძლება ვიცოდეთ მხოლოდ მატრიცა, y სვეტის გარდა. მისი ფორმირებისათვის ჩვენ გამოვიყენებთ ე.წ ლოგიკურ სქემას, რომელსაც შესასვლელზე მიეწოდება 0 ები და 1 ები, ხოლო გამოსასვლელზე იღებს 0 ს ან 1 ს, კონკრეტული ლოგიკური გამოსახულებიდან გამომდინარე. ასეთი სქემის შერჩევა დამოკიდებულია ოპერანდების რაოდენობაზე,

მაგალითისთვის შეგვიძლია მოვიყვანოთ ჩვენს მიერ შემუშავებული ლოგიკური სქემა, რომელიც განსაზღვრულია 12 ოპერანდზე:



სურ. 42 შემუშავებული ლოგიკური სქემა

მოცემულ სქემაში გამოყენებულია 4 ტიპის ატომური ოპერატორი: "და" (&) , "ან" (V), "უარყოფა" (˜), "ბიტური ან" (XOR). შესასვლელზე მიეწოდება 0 ები და 1 ები, საბოლოოდ კი მივიღებთ 0 ს ან 1 ს ამ სქემიდან გამომდინარე. შეგვიძლია მოვიყვანოთ კონკრეტული მაგალითი:



სურ. 43 ლოგიკური სქემის გამოყენების მაგალითი

შემავალი ინფორმაციის გარდა არანაკლებ მნიშვნელოვანია გამომავალი ინფორმაციის ნორმალიზაცია/დენორმალიზაცია. აუცილებელია წინასწარ იყოს ცნობილი რა ტიპის უნდა იყოს გამომავალი ინფორმაცია. ასეთი ტიპები შეიძლება იყოს: ლოგიკური ტიპი (0 ან 1), რიცხვითი ტიპი, რაიმე სხვა ობიექტური ტიპი. ხშირ შემთხვევაში საჭიროა ლოგიკური ტიპის გამომავალი ინფორმაცია. ვთქვათ მოცემულია d კოეფიციენტი (რიცხვითი სახის) და უნდა ვიპოვოთ რაიმე r_i^j ელემენტები (i=0,1....N, j=1....M) მისთვის შეგვიძლია გამოვიყენოთ შემდეგი მიმართებითი დამოკიდებულება:

$$r_i^j = \begin{cases} 0 \; if \;\; r_i^j \leq d \\ 1 \; if \;\; r_i^j > d \end{cases}$$

აღნიშნულ დამოკიდებულებაში საქმე გვაქვს ორი შესაძლო მნიშვნელობის მიღებასთან (0 ან 1) თუმცა იმავე გამომავალი ინფორმაციის ბინარული სახით ფორმირებისთვის შეგვიძლია გამოვიყენოთ ლოგიკური მიმართების უფრო კომპლექსური მეთოდები, მაგალითად რაიმე ლოგიკური ფორმულა ((A&B)||(A&C)). ასეთი კომპლექსური ლოგიკური ფორმულა შეიძლება მოიცავდეს ერთზე მეტ ოპერანდს და სხვადასხვა მიმართებებს მათ შორის.

რიცხვითი ტიპის შემთხვევაშიც შეიძლება გამოყენებულ იქნას მიმართება იგივე d კოეფიციენტთან მიმართებაში, მაგრამ ეს მიდგომა კარგი იქნება იმ შემთხვევაში თუ მისაღები რიცხვები არის წინასწარ განსაზღვრული სიმრავლის ელემენტები (მაგ. თუ სიმრავლეში არის რიცხვები {1,2,3,7,12}, მიმართებით მიღებული რიცხვიც უნდა იყოს ამ სიმრავლის ელემენტი). იმ შემთხვევაში თუ რიცხვის მნიშვნელობა შეიძლება იყოს ნებისმიერი, აუცილებელია განისაზღვროს ფორმულა, რომელიც მოგვცემს ოპტიმალურ მნიშვნელობას. ასეთი ფუნქციის განსაზღვრა დამოკიდებულია იმაზეც ეს მიღებული ინფორმაცია როგორი ტიპის/კონკრეტულად რომელი ალგორითმისთვის იქნება გამოყენებული როგორც შემავალი ინფორმაცია, რასაც აქ არ განვიხილავთ (რადგან აღნიშნული საკითხი წარმოადგენს ცალკე კვლევის საგანს).

განვიხილოთ კონკრეტული მაგალითი. ვთქვათ მოცემულია მონაცემები რომლებიც შეიცავენ ორ ცვლადზე რაიმე რიცხვითი ტიპის ინფორმაციას [10]:

ინდექსი	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

სურ. 44 საწყისი მონაცემები

მოცემული მონაცემები დავაჯგუფოთ ორ კლასტერად, პირველად ავიღოთ 1 და მე 4 ინდექსის მქონე სტრიქონები და ვიპოვოთ ცენტროიდები (გამოვიყენოთ ევკლიდეს ფორმულა):

	ინდექსი	ცენტროიდი
ჯგუფი 1	1	(1.0, 1.0)
ჯგუფი 2	4	(5.0, 7.0)

სურ. 45 საწყისი კლასტერები

თითოეული კლასტერული ჯგუფისთვის განვსაზღვროთ ინდექსების მიმდევრობები და გამოვთვალოთ ცენტროიდები (ისევ ევკლიდეს მეთოდით):

	კლასტერული ჯგუფი 1		კლასტერული ჯგუფი 2	
ბიჯი	ინდექსები	ცენტროიდი	ინდექსები	ცენტროიდი
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

სურ. 46 კლასტერების ასაგებად მონაცემთა განაწილება

საწყისი კლასტერული განაწილება შეიცვალა, აქედან გამომდინარე გვექნება:

	ინდექსი	ცენტროიდი
კლასტერი 1	1, 2, 3	(1.8, 2.3)
კლასტერი 2	4, 5, 6, 7	(4.1, 7.0)

სურ. 47 კლასტერის გარდაქმნა

ამის შემდეგ საბოლოოდ არ ვართ დარწმუნებულნი რომ თითოეული ობიექტი სწორ კლასტერულ ჯგუფშია, ამიტომ ვადარებთ თითოეული ინდექსის შესაბამისი ობიექტის საკუთარ კლასტერთან მანძილი ნაკლებია თუ არა მეორე კლასტერთან მანძილისა და თუ არა (ე.ი უფრო ახლოსაა მეორე კლასტერთან) უნდა გადავიტანოთ მეორე კლასტერში.

ინდექსი	მანძილი პირველ კლასტერამდე	მანძილი მეორე კლასტერამდე
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

სურ. 48 მანძილების დათვლა სხვადასხვა კლასტერამდე

საბოლოო ჯამში თითოეული ინდექსის შესაბამისი ობიექტი მოხვდება იმ კლასტერში რომელთანაც უფრო ახლოსაა (მანძილი შესაბამის კლასტერამდე არის ნაკლები ვიდრე სხვა კლასტერამდე). აქედან გამომდინარე გვექნება:

	ინდექსი	ცენტროიდი
კლასტერი 1	1, 2	(1.3, 1.5)
კლასტერი 2	3, 4, 5, 6, 7	(3.9, 5.1)

სურ. 49 საბოლოოდ მიღებული კლასტერი

საბოლოო ჯამში იტერაციული გზით მივედით ოპტიმალურ განაწილებამდე, თუმცა არის შემთხვევები როდესაც ასეთი იტერაციების რაოდენობა ან ძალიან დიდია, ან უსასრულოდ გრძელდება. ასეთ შემთხვევაში შეგვიძლია დავეყრდნოთ იტერაციათა კონკრეტულ რაოდენობას, რომელიც მიახლოებით ან ზუსტად არის განსაზღვრული კონკრეტული ამოცანისთვის.

გამომავალი ინფორმაცია საბოლოოდ არის ინდექსების შესაბამისი ობიექტები. ამ შემთხვევაში ვიცით, რომ პირველ კლასტერში არის 1 და 2- ე ინდექსის შესაბამისი ობიექტები, ხოლო დანარჩენი ეკუთვნის მეორე

კლასტერს. იმისათვის რომ აღნიშნული მონაცემები გადავიყვანოთ ბინარულ ფორმატში, განვსაზღვროთ მიმართება d კოეფიციენტის მონაწილეობით, კერძოდ:

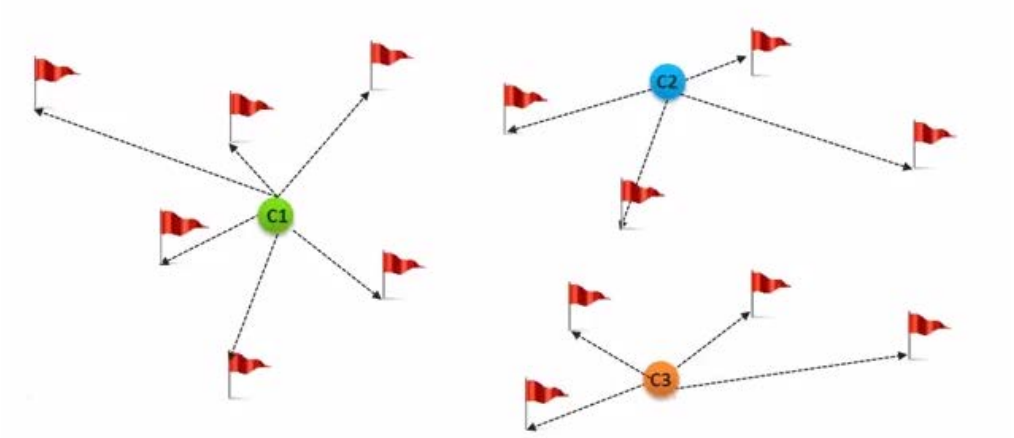
$$r_i^j = \begin{cases} 0 \; if \;\; r_i^j \leq d \\ 1 \; if \;\; r_i^j > d \end{cases}$$

თუ d=1.1 მაშინ:

ინდექსი	A	B
1	0	0
2	1	1

სურ. 50 საბოლოოდ მიღებული გარდაქმნა ბინარულ ფორმატში

შეიძლება მოვახდინოთ სხვადასხვა სახის ვიზუალიზაცია აგებული კლასტერების მასთან დაკავშირებული კვანძებით:



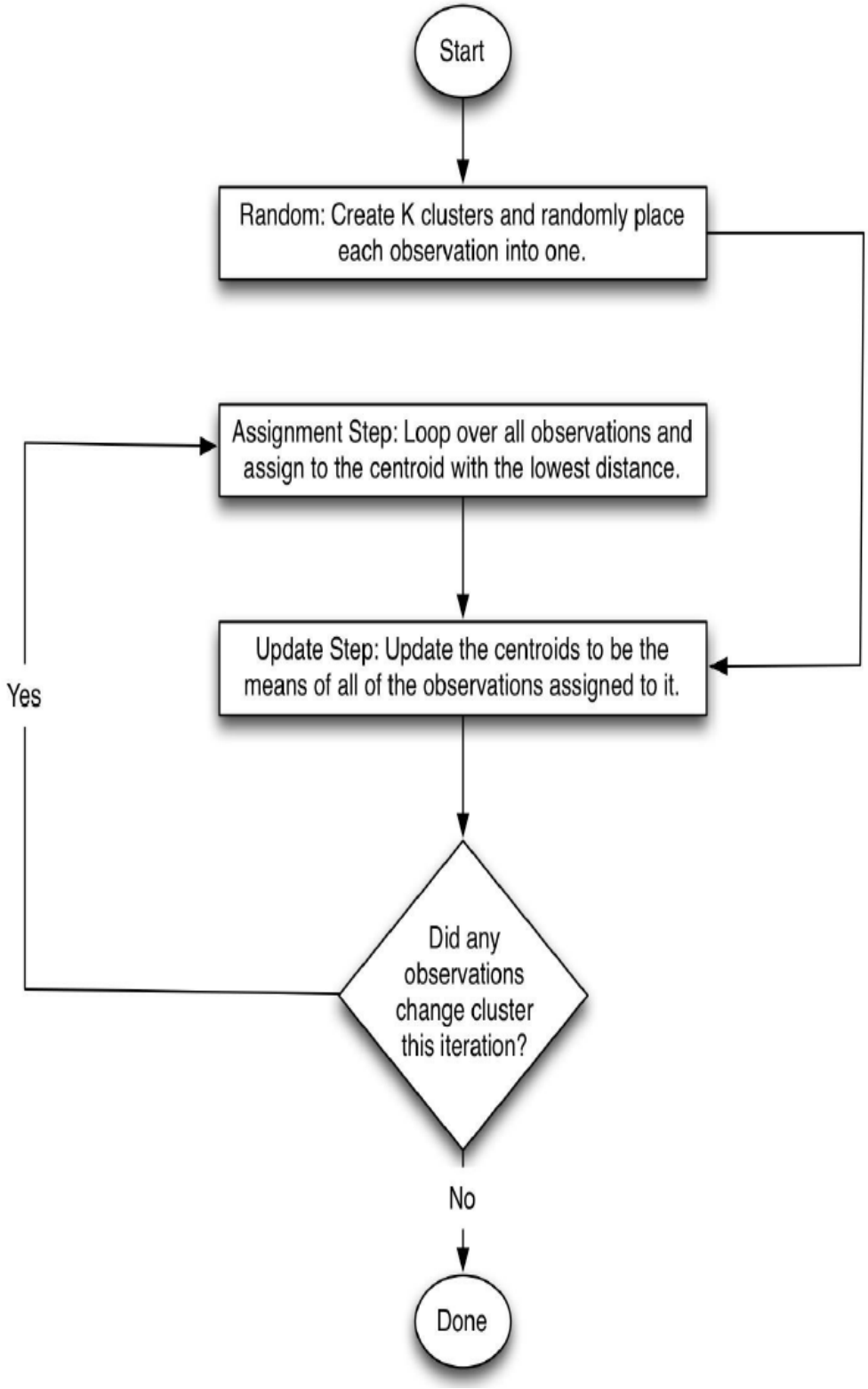
სურ. 51 კლასტერიზაციის ერთ ერთი ვიზუალიზაცია

არსებობს კლასტერების ინიციალიზაციის ორი ძირითადი ვარიანტი“

- **შემთხვევითი ინიციალიზაცია** - ვიღებთ დაკვირვებათა სიმრავლეს და შემთხვევითად ვანაწილებთ წერტილებს სხვადასხვა კლასტერში;

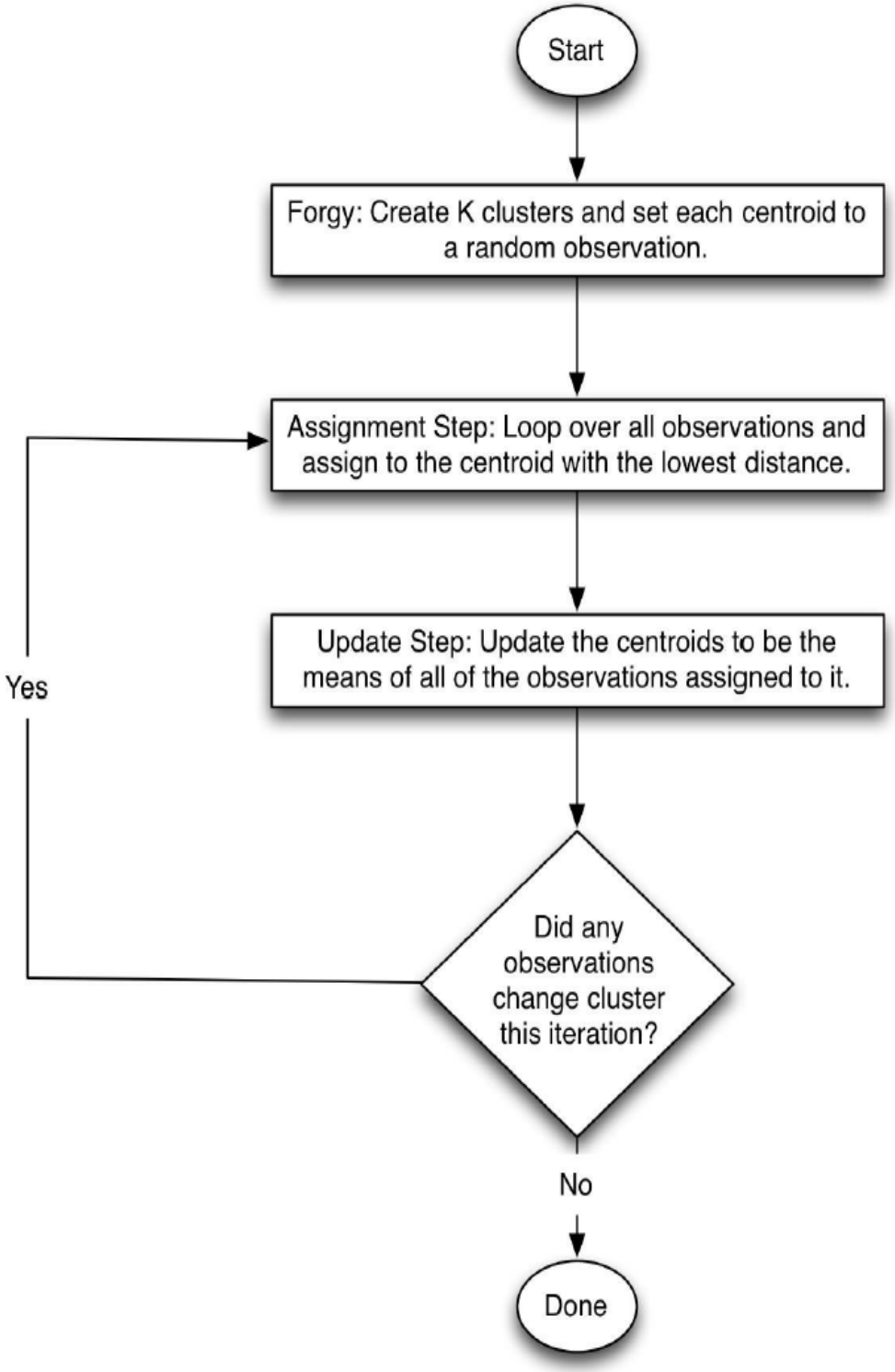
- **ცენტროიდზე დაფუძვნებული** - ვპოულობთ ცენტროიდებს და წერტილებს ვათავსებთ იმ კლასტერში, რომლის შესაბამისი ცენტროიდიც უფრო ახლოს არის ამ წერტილთან;

შემთხვევითი ინიციალიზაციის ბლოკ სქემა შემდეგნაირად გამოიყურება



სურ. 52 შემთხვევითი ინიციალიზაციის სქემა

ხოლო ცენტროიდებზე დაფუძვნებული ინიციალიზაცია კი შემდეგნაირად გამოიყურება:



სურ. 53 ცენტროიდებზე დაფუძვნენული ინიციალიზაცია

მონაცემთა ნორმალიზაცია შეიძლება იყოს საკმაოდ რთული პროცესი და თითოეული ტიპის ალგორითმისათვის საჭირო გახდეს ცალკე ნორმალიზაციის ალგორითმის მოფიქრება. როგორი ტიპის მონაცემებიც არ

უნდა გვქონდეს თუკი მათ გარდავქმნით რიცხვებში შემდეგ უფრო მარტივი იქნება ნორმალიზაცია/დენორმალიზაცია.

ვთქვათ მოცემულია რაიმე რიცხვითი შუალედი და ასევე ნორმალიზებული შუალედი:

- dataHigh - საწყისი მოცემული მნიშვნელობის მაქსიმუმი;
- dataLow - საწყისი მოცემული მნიშვნელობის მინიმუმი;
- normalizedHigh - ნორმალიზებული მნიშვნელობების მაქსიმუმი;
- normalizedLow - ნორმალიზებული მნიშვნელობების მინიმუმი;

პირობითად განვსაზღვროთ რაიმე რიცხვითი მნიშვნელობები, რომლის ნორმალიზაცია/დენორმალიზაციასაც შემდეგ მოვახდენთ:

- dataHigh - 4000;
- dataLow - 100;
- normalizedHigh 1;
- normalizedLow -1;

უნდა გამოვთვალოთ ე.წ dataRange და normalizedRange:

dataRange = dataHigh – dataLow = 4000 – 100 = 3800;

normalizedRange = normalizedHigh – normalizedLow = 1 – (-1) = 2;

დავუშვათ მოცემულია რაიმე მიმდინარე sample=1000 მნიშვნელობა, მის საფუძველზე კი ვითვლით:

$$d = samle - dataLow = 1000 - 100 = 900;$$

მიღებული მნიშვნელობა გარდავქმნათ პროცენტად:

$$dPct = d / dataRange = 900 / 3800 = 0.230769\ldots\ (23\%)$$

საბოლოოდ ვითვლით:

$$dNorm = dRange * dPct = 2 * 0.23 = 0.46$$

საბოლოოდ მიღებული მნიშვნელობა შეგვიძლია დავუმატოთ normalizedLow ს:

$$Normalized = normalizedLow + dNorm = -1+0.46 = -0.54;$$

მთლიანი პროცესი შეგვიძლია აღვწეროთ ერთი განტოლებით:

$$f(x) = \frac{(x-d_L)(n_H-n_L)}{(d_H-d_L)} + n_L$$

სურ. 54 ნორმალიზაციის ერთიანი ფორმულა

ახლა კი ვნახოთ როგორ ხდება დენორმალიზაცია (იგივე მაგალითის საფუძველზე)

```
-0.54 - (-1) = 0.46

0.46 / 2 = 0.23

0.23 * 3900 = 897

897 + dataLow = 897 + 100 = 997
```

$$f(x) = \frac{(d_L-d_H)x-(n_H \cdot d_L)+d_H \cdot n_L}{(n_L-n_H)}$$

სურ. 55 დენორმალიზაცია (მთლიანი პროცესი)

აღნიშნული მეთოდოლოგია წარმოადგენს ერთ-ერთ ვარიანტს თუ როგორ შეიძლება ერთი ალგორითმის შედეგების გამოყენებით ინფორმაცია მიეწოდოს მეორე ალგორითმს (თუ სისტემა ისეთნაირადაა მოწყობილი, რომ ერთი ალგორითმი დამოკიდებულია მეორეზე და ა.ს). რა თქმა უნდა შესაძლებელია შემავალი და გამომავალი ინფორმაციის ფორმირების განსხვავებული მეთოდოლოგიის შემუშავება, მაგრამ აღნიშნული მიდგომა წარმოადგენს მცდელობას ინტელექტუალურ სისტემაში (სადაც გამოყენებულია მანქანური სწავლების ალგორითმები) დაიხვეწოს სისტემის წარმადობა და გახდეს უფრო ეფექტური და სრულყოფილი.

შეგვიძლია მოვიყვანოთ გარკვეული შედარება კლასიფიკაციისა და კლასტერიზაციის ალგორითმებს შორის:

Classification vs Clustering

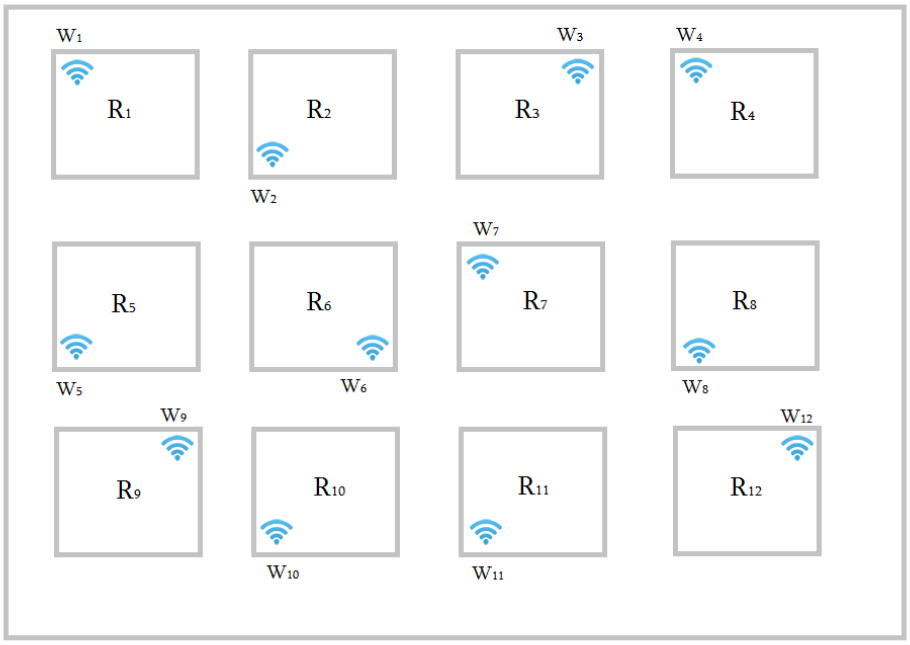
Criteria	Classification	Clustering
Prior Knowledge of classes	Yes	No
Use case	Classify new sample into known classes	Suggest groups based on patterns in data
Algorithms	Decision Trees, Bayesian classifiers	K-means, Expectation Maximization
Data Needs	Labeled samples from a set of classes	Unlabeled samples

სურ. 56 კლასიფიკაციისა და კლასტერიზაციის შედარება

განხილული ალგორითმების გამოყენება შესაძლოა იყოს ეფექტური თუ მოვახდენთ მათ კომბინაციას. კომბინაცია არ გულისმობს ალგორითმების შერწყმას და რაიმე ახალი ალგორითმის მიღებას, არამედ ის ძირითადად ეფუძვნება გარკვეულ პრინციპს, რომლის მიხედვითაც მოდელმა იცის როგორი ტიპის ალგორითმები უნდა ამოარჩიოს ამა თუ იმ ამოცანის გადაწყვეტის დროს.

4. ექსპერიმენტი კონკრეტულ მონაცემებზე

ვთქვათ მოცემულია შენობის რაიმე ნაწილი სადაც განლაგებულია 12 ოთახი , რომელთათვისაც დაყენებულია 12 WIFI მოწყობილობა (თითო WIFI მოწყობილობა აყენია თითო ოთახში). თითოეული ამ ოთახისათვის მოცემული გვაქვს 16 სხვადასხვა წერტილი:



სურ. 57 Wifi ს განლაგების კონკრეტული მაგალითი

შემოვიტანოთ აღნიშვნები:

- $W_1 \ldots W_{12}$ - Wifi მოწყობილობები;
- $R_1 \ldots R_{12}$ - ოთახები;
- r_i^j დაჭერის/კავშირის სიმძლავრე, სადაც i განსაზღვრავს ოთახის ინდექსს, ხოლო j აღნიშნულ ოთახში წერტილის ინდექსს (რომელი წერტილიდანაც მეტ-ნაკლებად ხდება Wifi მოწყობილობასთან დაკავშირება);

r_i^j პარამეტრი განისაზღვრება შემდეგნაირად:

$$r_i^j = \begin{cases} 0 \text{ თუ } r_i^j \leq d \\ 1 \text{ თუ } r_i^j > d \end{cases}$$

d წარმოადგენს წინასწარ მოცემულ რაიმე კოეფიციენტს. ამ კონკრეტული მაგალითის შემთხვევაში d = 0.67;

პირველ ეტაპზე ვიყენებთ კლასტერიზაციის ალგორითმს, კერძოდ K-means ალგორითმს რათა ვიპოვოთ შემდეგი სიმრავლის მნიშვნელობები (r_i^j ის საწყისი მნიშვნელობები, ხოლო შემდეგ ამ რიცხვით მნიშვნელობებს დავიყვანთ 0 ზე ან 1 ზე, როგორც ეს ზემოთ აღწერილ ფორმულაშია):

$R_1 = \{r_1^1, r_1^2, r_1^3, r_1^4, r_1^5, r_1^6, r_1^7, r_1^8, r_1^9, r_1^{10}, r_1^{11}, r_1^{12}, r_1^{13}, r_1^{14}, r_1^{15}, r_1^{16}\}$;

$R_2 = \{r_2^1, r_2^2, r_2^3, r_2^4, r_2^5, r_2^6, r_2^7, r_2^8, r_2^9, r_2^{10}, r_2^{11}, r_2^{12}, r_2^{13}, r_2^{14}, r_2^{15}, r_2^{16}\}$;

$R_3 = \{r_3^1, r_3^2, r_3^3, r_3^4, r_3^5, r_3^6, r_3^7, r_3^8, r_3^9, r_3^{10}, r_3^{11}, r_3^{12}, r_3^{13}, r_3^{14}, r_3^{15}, r_3^{16}\}$;

$R_4 = \{r_4^1, r_4^2, r_4^3, r_4^4, r_4^5, r_4^6, r_4^7, r_4^8, r_4^9, r_4^{10}, r_4^{11}, r_4^{12}, r_4^{13}, r_4^{14}, r_4^{15}, r_4^{16}\}$;

$R_5 = \{r_5^1, r_5^2, r_5^3, r_5^4, r_5^5, r_5^6, r_5^7, r_5^8, r_5^9, r_5^{10}, r_5^{11}, r_5^{12}, r_5^{13}, r_5^{14}, r_5^{15}, r_5^{16}\}$;

$R_6 = \{r_6^1, r_6^2, r_6^3, r_6^4, r_6^5, r_6^6, r_6^7, r_6^8, r_6^9, r_6^{10}, r_6^{11}, r_6^{12}, r_6^{13}, r_6^{14}, r_6^{15}, r_6^{16}\}$;

$R_7 = \{r_7^1, r_7^2, r_7^3, r_7^4, r_7^5, r_7^6, r_7^7, r_7^8, r_7^9, r_7^{10}, r_7^{11}, r_7^{12}, r_7^{13}, r_7^{14}, r_7^{15}, r_7^{16}\}$;

$R_8 = \{r_8^1, r_8^2, r_8^3, r_8^4, r_8^5, r_8^6, r_8^7, r_8^8, r_8^9, r_8^{10}, r_8^{11}, r_8^{12}, r_8^{13}, r_8^{14}, r_8^{15}, r_8^{16}\}$;

$R_9 = \{r_9^1, r_9^2, r_9^3, r_9^4, r_9^5, r_9^6, r_9^7, r_9^8, r_9^9, r_9^{10}, r_9^{11}, r_9^{12}, r_9^{13}, r_9^{14}, r_9^{15}, r_9^{16}\}$;

$R_{10} = \{r_{10}^1, r_{10}^2, r_{10}^3, r_{10}^4, r_{10}^5, r_{10}^6, r_{10}^7, r_{10}^8, r_{10}^9, r_{10}^{10}, r_{10}^{11}, r_{10}^{12}, r_{10}^{13}, r_{10}^{14}, r_{10}^{15}, r_{10}^{16}\}$;

$R_{11} = \{r_{11}^1, r_{11}^2, r_{11}^3, r_{11}^4, r_{11}^5, r_{11}^6, r_{11}^7, r1_1^8, r_{11}^9, r_{11}^{10}, r_{11}^{11}, r_{11}^{12}, r_{11}^{13}, r_{11}^{14}, r_{11}^{15}, r_{11}^{16}\}$;

$R_{12} = \{r_{12}^1, r_{12}^2, r_{12}^3, r_{12}^4, r_{12}^5, r_{12}^6, r_{12}^7, r_{12}^8, r_{12}^9, r_{12}^{10}, r_{12}^{11}, r_{12}^{12}, r_{12}^{13}, r_{12}^{14}, r_{12}^{15}, r_{12}^{16}\}$;

$Y = \{y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11}, y_{12}, y_{13}, y_{14}, y_{15}, y_{16}\}$;

y_i მნიშვნელობები კი განისაზღვრება შემდეგი ფორმულის საფუძველზე:

$$y_i = \begin{cases} 0 \text{ თუ } y_i = d; \\ 1 \text{ თუ } y_i \neq d; \end{cases}$$

K-means ალგორითმის საფუძველზე ზემოაღნიშნულ სიმრავლეში მივიღებთ შემდეგ მნიშვნელობებს:

$R_1 = \{1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1\}$;

$R_2 = \{0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0\}$;

$R_3 = \{1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0\}$;

$R_4 = \{1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1\}$;

R_5 = {0 , 1 , 0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 1 , 0 , 1 , 0 , 0 , 1};

R_6 = {1 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 1 , 1 , 1 , 0 , 1 , 0 , 1};

R_7 = {0 , 0 , 1 , 0 , 1 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 1 , 1 , 1 , 0};

R_8 = {1 , 0 , 1 , 1 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 1 , 1 , 0};

R_9 = {0 , 1 , 0 , 0 , 0 , 1 , 1 , 0 , 1 , 1 , 1 , 1 , 0 , 0 , 0 , 1};

R_{10}= {1 , 0 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 1 , 1 , 0 , 1 , 0 , 1};

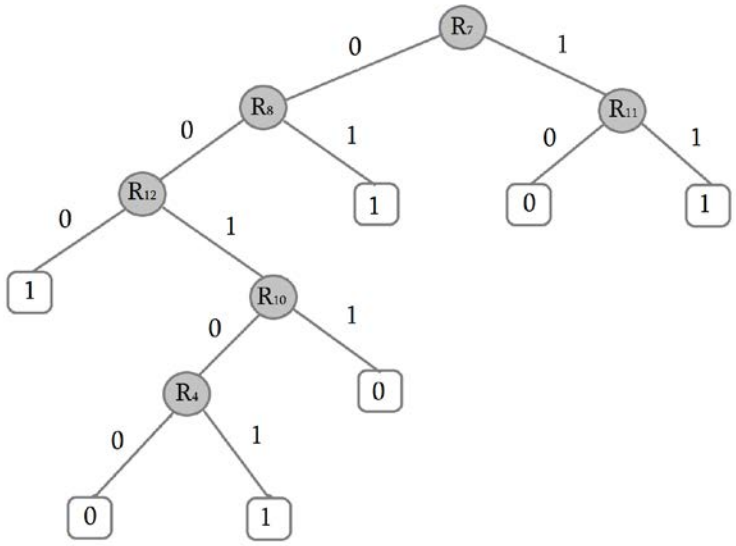
R_{11}= {1 , 1 , 0 , 1 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0};

R_{12}= {1 , 1 , 1 , 0 , 0 , 0 , 1 , 1 , 1 , 1 , 0 , 1 , 1 , 1 , 1 , 0};

Y = {1 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 0 , 1 , 1 , 0 , 1 , 0 , 0 , 1};

თითოეული R_i წარმოადგენს მატრიცის ვექტორ-სვეტს, ხოლო Y ვექტორ სვეტი კი წარმოადგენს ვექტორ სვეტებისაგან მიღებული მატრიცის სტრიქონის შესაბამისობას . ინფორმაციის ასეთი სახით წარმოდგენა აუცილებელია Decision Tree ს ალგორითმისთვის, კერძოდ მისი შემავალი ტიპის ინფორმაცია წარმოდგება ზუსტად ასეთი სახით.

საბოლოო ეტაპზე გამოვიყენოთ Decision Tree ს ალგორითმი (შემავალი ინფორმაცია კი იქნება ზემოაღნიშნული მატრიცა) რის შედეგადაც მივიღებთ შემდეგ გადაწყვეტილებათა ხეს:



სურ. 58 ექსპერიმენტის შედეგად მიღებული გადაწყვეტილებათა ხე

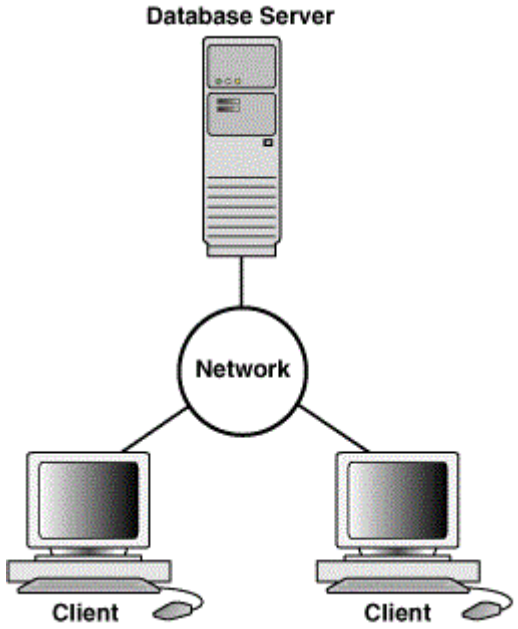
5. სისტემის პროგრამული მოდელის სახით წარმოდგენა

ნაშრომში ვისაუბრეთ ინტელექტუალური სისტემის სხვადასხვა კომპონენტებზე, რომელთაც თავიანთი მკვეთრად განსაზღვრული ფუნქციონალობა აკისრიათ. ხაზი გავუსვით იმ ფაქტს, რომ მანქანური სწავლება არის მისი ერთ-ერთი უმთავრესი ნაწილი.

ინტელექტუალური სისტემა შესაძლოა განხილულ იქნას როგორც პროგრამა, რომელიც სხვა ჩვეულებრივი პროგრამებისაგან განსხვავებით გამოირჩევა დასწავლის უნარით და აზროვნებს გარკვეულ დონეზე, სწავლობს გამოცდილებიდან და აქვს უნარი გააუმჯობესოს საკუთარი თავი. სანამ უშუალოდ ინტელექტუალური სისტემის ჩვენს მიერ შეთავაზებულ არქიტექტურაზე გადავალთ, ვნახოთ თუ რას წარმოადგენს **N დონიანი** არქიტექტურის მქონე პროგრამული უზრუნველჰყოფები. თანამედროვე სამყაროში ძირითადად მაინც გამოყოფილია სამ დონიანი არქიტექტურის მქონე აპლიკაციები, რომლის მოდიფიკაციითაც შესაძლოა მივიღოთ უფრო მეტი „დონეები“ (რასაკვირველია ამოცანის სწორ დასმასა და პროგრამისტის კვალიფიკაციაზე არის დამოკიდებული რამდენად სწორად იქნება შექმნილი და განვითარებული ესა თუ ის პროგრამული უზრუნველჰყოფა).

მე 20 საუკუნის ბოლოს პოპულარული იყო ორ დონიანი არქიტექტურის მქონე პროგრამული უზრუნველჰყობა. პირველ დონეზე წარმოდგენილი იყო პროგრამის ვიზუალური მხარე (რაზეც მომხმარებელი მუშაობდა უშუალოდ), ხოლო მეორე მხარეს კი შეიძლება ყოფილიყო მონაცემთა ბაზა. გამოთვლების ნაწილი სრულდებოდა მომხმარებლის მხარეს, ხოლო ნაწილი კი მონაცემთა ბაზის მხარეს. ასეთ დროს პროგრამისტს მეტწილად უწევდა იმაზე ფიქრი როგორ გადაენაწილებინა გამოთვლები ისეთნაირად, რომ მომხმარებლის კომპიუტერი ზედმეტად არ ყოფილიყო დატვირთული. ასევე ისიც გასათვალისწინებელი იყო თუ რომელი ოპერაციული სისტემა იყო დაყენებული მომხმარებლის კომპიუტერზე და ჰქონდა თუ არა თავსებადობა პროგამას აღნიშნულ

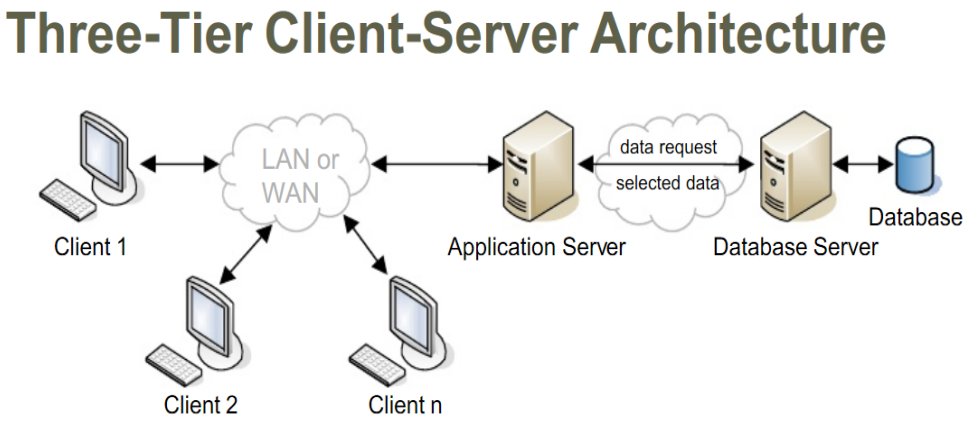
ოპერაციულ სისტემასთან. ასეთი ტიპის დეტალები საბოლოო ჯამში ქმნიდა საკმაოდ კომპლექსურ სისტემებს, სადაც შეცდომების პოვნა და გასწორება დიდ სირთულეებთან იყო დაკავშირებული.



სურ. 59 ორ დონიანი არქიტექტურის მაგალითი

21 ე საუკუნის დასაწყისში ვების განვითარებასთან ერთად ფეხი მოიკიდა სამ დონიანი არქიტექტურის პროგრამულმა უზრუნველჰყოფის განვითარებამაც, რომლის ძირითადი რგოლებია: **კლიენტი , სერვერი და ბაზა.** რასაკვირველია კლიენტი წარმოადგენს ბრაუზერს, რომლის საშუალებითაც მომხმარებელი მუშაობს პროგრამაში. ეს დადებით მხარედ შეიძლება ჩაითვალოს რადგან ასეთ შემთხვევაში საერთოდ არ აქვს არანაირი მნიშვნელობა რომელ ოპერაციულ სისტემაზე მუშაობს მომხმარებელი, რომელი ვერსიის ოპერაციული სისტემა უყენია და ა.შ. ერთადერთი მნიშვნელოვანია ის თუ რომელი ბრაუზერით მუშაობს პროგრამაში. ეს უკანასკნელი საკმაოდ მნიშვნელოვანია რადგან ხშირად ასეთი ტიპის პროგრამულ უზრუნველჰყოფას შესაძლოა არ გააჩნდეს ყველა ბრაუზერის მხარდაჭერა. ეს ფაქტიც გარკვეულწილად ასახავს პროგრამისტის კვალიფიკაციასა და პროფესიონალიზმს, თუმცა არსებობს შემთხვევები, რომლის საფუძველზეც გამართლებულია ის ფაქტი, რომ მოცემულ პროგრამას არ გააჩნდეს ყველა ბრაუზერის მხარდაჭერა (ასეთი

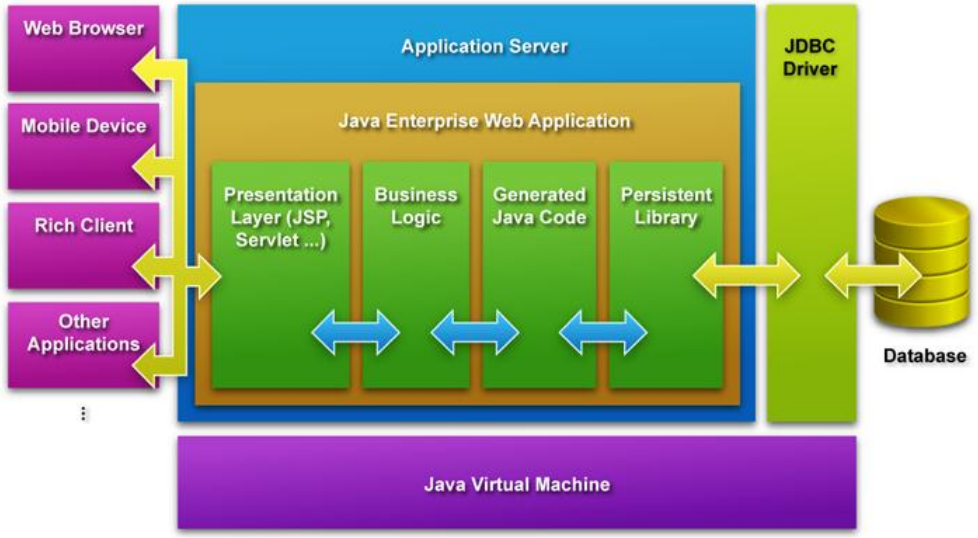
ფაქტორები შეიძლება იყოს საჭიროზე მცირე დრო, რომელიც პროგრამისტმა დახარჯა სისტემის შექმნაზე და ა.შ).



სურ. 60 სამ დონიანი არქიტექტურა

- **კლიენტის დონე** [11] - წარმოადგენს რაიმე ინტერფეისს , რომლის დახმარებითაც მუშაობს მომხმარებელი სისტემაში. მას შესაძლოა საერთოდ არ ჰქონდეს წარმოდგენა აპლიკაციის დანარჩენ ნაწილზე, უბრალოდ იცის როგორ გამოიყენოს პრორამა. კლიენტის ინტერფეისი შესაძლოა აწყობილი იყოს როგორც ბრაუზერის მხარეს ასევე ფანჯრების სისტემით (ამ საკითხს წყვეტს შესაბამისი პროგრამის არქტიქტორი სხვებთან შეთანხმებით);
- **სერვერის დონე** [12] - სერვერი წარმოადგენს მძლავრ კომპიუტერს, რომელსაც შეუძლია პარალელურად გადაჭრას 1 ზე მეტი სხვადასხვა სირთულის ამოცანები. სერვერის დონეზე არ იგულისხმება მხოლოდ ზემოთთქმული განმარტება. პროგრამულ უზრუნველჰყოფაში უმეტესად გვხვდება აპლიკაციის სერვერი [14], რომელზეც უშუალოდ მუშაობს პროგრამის ბირთვი, ხოლო აპლიკაციის სერვერი კი გაშვებულია რაიმე ფიზიკურ მანქანაზე, რომელსაც ფიზიკურ სერვერსაც უწოდებენ. საბოლოო ჯამში გვაქვს ფიზიკური სერვერი (შეიძლება სერვერი იყოს ვირტუალურიც თუმცა აღნიშნული ვირტუალური სერვერები საბოლოო ჯამში მაინც გაშვებულია რომელიმე ფიზიკურ სერვერზე), რომელზეც აყენია რაიმე სერვერული ოპერაციული სისტემა [15] და მუდმივად გაშვებულია

ერთი ან რამოდენიმე აპლიკაციის სერვერი. აპლიკაციის სერვერი კი მოიცავს სხვადასხვა ტექნოლოგიებს და თავად აპლიკაციებს, რომელსაც ამუშავებს აღნიშნული აპლიკაციის სერვერი. უფრო ზუსტი წარმოდგენისთვის შეგვიძლია მოვიყვანოთ შესაბამისი დიაგრამა:



სურ. 61 ერთ-ერთი აპლიკაციის სერვერვის მაგალითი

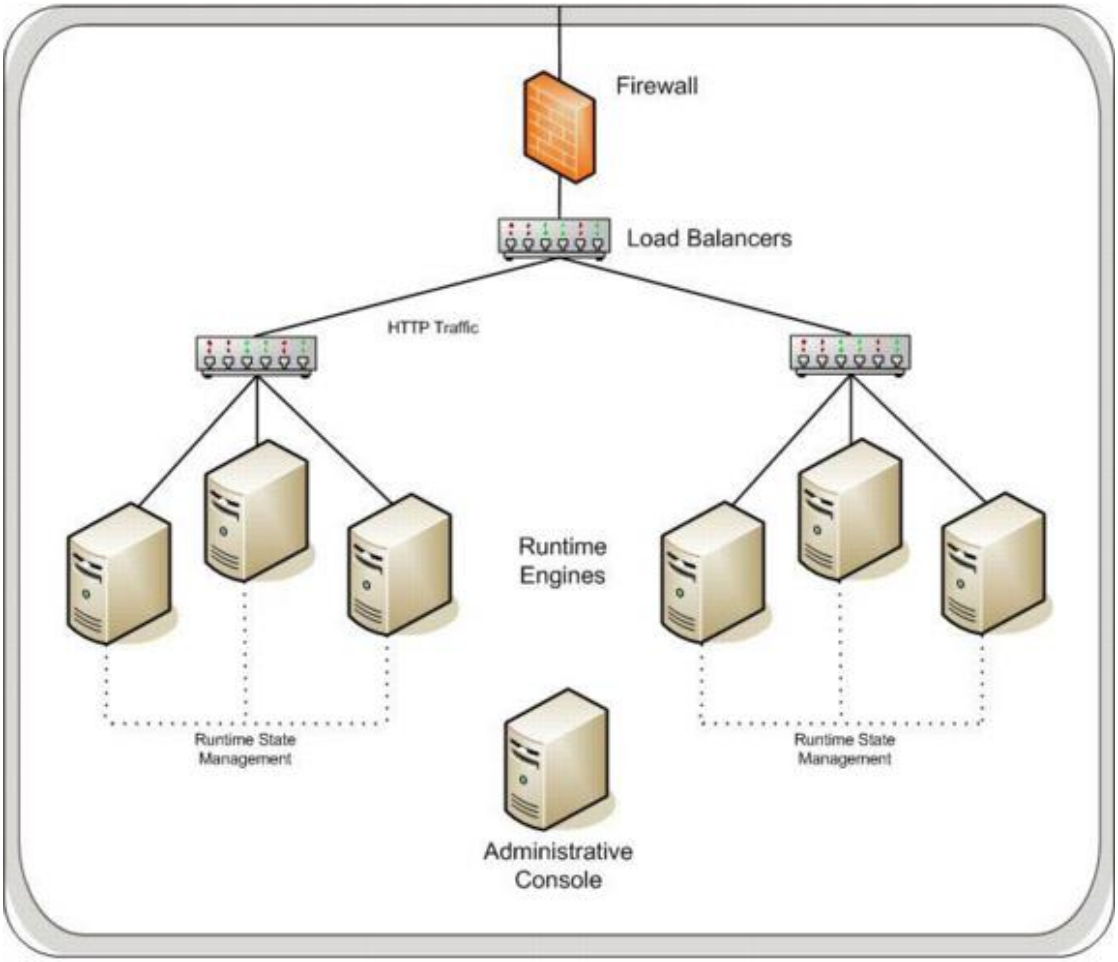
- **მონაცემთა ბაზის დონე** [13] - მონაცემთა ბაზა წარმოადგენს გარკვეული ლოგიკითა და სტრუქტურით დაჯგუფებულ ინფორმაციას. აღნიშნულ ინფორმაციას შესაძლოა იყენებდეს ერთი ან მეტი აპლიკაცია აქედან გამომდინარე ძალიან დიდი მნიშვნელობა აქვს თუ როგორ არის გაკეთებული მონაცემთა ბაზა, რა დატვირთვას შეუძლია რომ გაუძლოს და ა.შ.

არსებობს აპლიკაციები, რომლებიც შესაძლოა ხასიათდებოდნენ სტანდარტულზე დიდი დატვითვით. ასეთი ტიპის აპლიკაციებში უდიდესი დატვირთვა მოდის როგორც სერვერზე ასევე მონაცემთა ბაზაზე. ამ პრობლემის თავიდან აცილება შესაძლებელია ორი გზით:

- გავზარდოთ სერვერის/ბაზის ფიზიკური რესურსები
- მოვახდინოთ კლასტერიზაცია

როდესაც დატვირთვა დიდია, მაგრამ პრობლემის გადაჭრა შესაძლებელია რესურსების გაზრდით, საბოლოო ჯამში ეს არც ისეთი დიდი პრობლემაა,

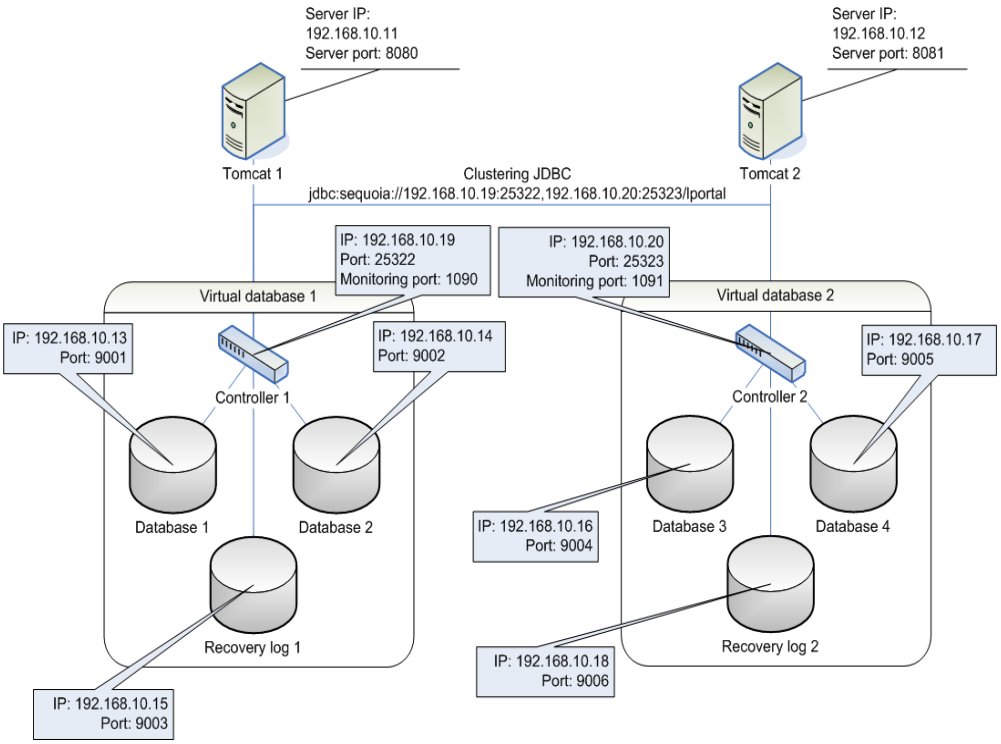
მაგრამ როდესაც მომხმარებელთა რაოდენობა იზრდება, ამ პრობლემას თავიდან ვერ ავიცილებთ მხოლოდ ფიზიკური რესურსების გაზრდით. ასეთ შემთხვევაში საჭირო ხდება სერვერის კლასტერის აგება, სადაც გაერთიანებული იქნება ორი ან მეტი სერვერი. სანამ მომხმარებლის მიერ გამოშვებული მოთხოვნა უშუალოდ რომელიმე სერვერს მიმართავს, საქმეში ჩაერთვება სპეციალური მოდული, რომელსაც ეწოდება loadbalancer ი და რომელიც არის შუამავალი კლიენტსა და სერვერს შორის:



სურ. 62 სერვერის კლასტერის მაგალითი

Loadbalancer ის მუშაობის პრინციპი ძირითად შემთხვევაში შემდეგია: მოთხოვნის შემოსვლისას ის აკვირდება ყველა სერვერს და მომხმარებელს გაუშვებს იმ სერვერზე, რომელზეც მიმდინარე მომენტში დატვირთვა ყველაზე ნაკლებია. რასაკვირველია მომხმარებელი ამას ვერ გრძნობს, მისთვის მთავარია იმუშაოს პროგრამაში შეუფერხებლად.

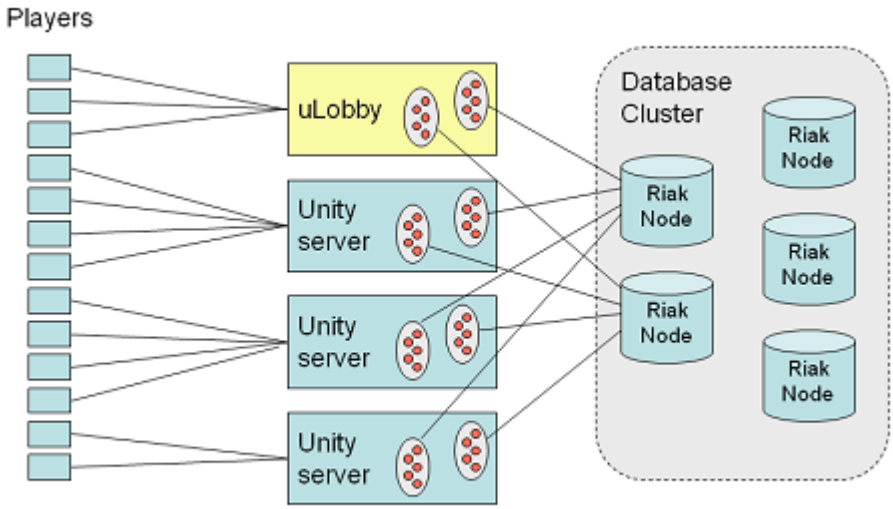
აბსოლიტურად იგივე პრინციპით შეგვიძლია განვიხილოთ მონაცემთა ბაზის კლასტერიზაციაც:



სურ. 63 მონაცემთა ბაზის კლასტერიზაცია

შესაძლოა სისტემა იმდენად რთული იყოს, რომ ის ხასიათდებოდეს როგორც სერვერული კლასტერით ასევე მონაცემთა ბაზის კლასტერით. ასეთი ტიპის არქიტექტურას გააჩნია მრავალი სხვადასხვა ვარიანტი, რომელთაგან ერთ-ერთია შემდეგი მიდგომა (განვიხილოთ სერვერის შემთხვევაში) : პროგრამა მიმართავს არა პირდაპირ მთავარ სერვერს, არამედ სპეციალურად გამოყოფილ სერვერს, რომელიც თავის მხრივ მიმართავს მთავარ სერვერს, დანარჩენი პროცესი კი უცვლელია. ეს ყოველივე შემოღებულია უსაფრთხოებიდან გამომდინარე, რადგან შესაძლოა მთავარ სერვერზე მოხდეს გარკვეული შეტევები და სენსიტიური ინფორმაციის მოპარვა. აღნიშნული მიდგომა გამოიყენება მონაცემთა ბაზების შემთხვევაშიც. არის ასევე არქიტექტურა, რომლის მიხედვითაც წარმოდგენილია ტრანზაქციების მონაცემთა ბაზა ცალკე, ხოლო უბრალოდ მონაცემების ბაზა ცალკე.

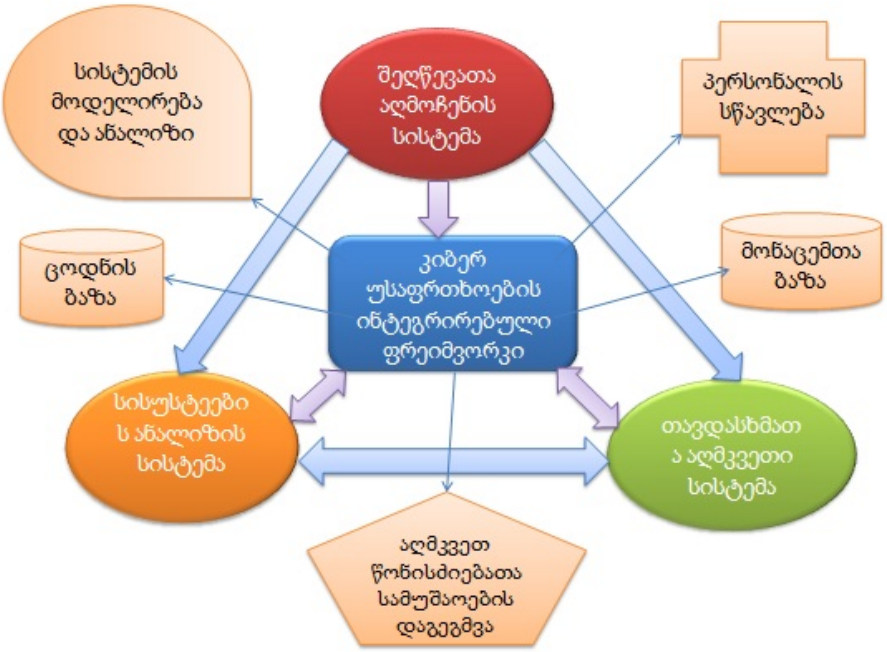
როგორც ვნახეთ არსებობს უამრავი სხვადასხვა ვარიანტი თუ როგორ შეიძლება მოხდეს აპლიკაციის არქიტექტურის მოფიქრება და აგება , ყოველივე ზემოთ თქმული შეგვიძლია გავაერთიანოთ ერთ დიაგრამად სადაც წარმოდგენილია როგორს სერვერის კლასტერი, ასევე მონაცემთა ბაზის კლასტერიც:



სურ. 64 სერვერისა და ბაზის კლასტერის ერთიანი მაგალითი

ჩვენს მიერ შემუშავებული მოდელი შეგვიძლია გამოვიყენოთ ისეთი ინტელექტუალური ტიპის სისტემის შესაქმნელად, რომელიც უზრუნველჰყობს კიბერ უსაფრთხოებას. ნაშრომში ჩვენ მოვიყვანეთ იმის მაგალითი თუ როგორ შეიძლება რაიმე სისტემა დავიცვათ (საუბარია დამატებითი მოდულის შექმნაზე რომელიც მონიტორინგს გაუწევს რაიმე სისტემას, დააფიქსირებს შემოჭრებს და აღკვეთს მას).

არსებული არქიტექტურა მოიცავს ისეთ ძირითად ნაწილებს როგორიცაა კიბერ უსაფრთხოების ინტეგრირებული ფრეიმვორკი, ცოდნის ბაზა და ა.შ. ეს ყველაფერი შეგვიძლია წარმოვადგინოთ შემდეგი სურათის სახით:



სურ. 65 ჩვენს მიერ შემუშავებული სისტემის ზოგადი არქიტექტურა

ძირითადი ბირთვი რომელმაც უნდა უზრუნველყოს მუშაობის გამართულობა არის კიბერ უსაფრთხოების ინტეგრირებული ფრეიმვორკი. იგი არის სისტემის ისეთი ნაწილი რომელსაც წვდომა აქვს სხვა ყველა დანარჩენ ნაწილზე ისეთებზე როგორიცაა : მონაცემთა ბაზა, აღმკვეთ ღონისძიებათა სამუშაოების დაგეგმვა, ცოდნის ბაზა და ა.შ.

ძირითადი მოქმედების პრინციპი გამოიხატება შემდეგში: პირველ ეტაპზე სისტემა „უსმენს“ ქსელის იმ ნაწილს სადაც შეიძლება მოხდეს შემოჭრა, არა სასურველი ინფორმაციის შემოსვლა და ა.შ (რაც მოიცავს საბრთხეს). შემდეგ ეტაპზე შეღწევათა აღმომჩენი სისტემა კიბერ უსაფრთხოების ინტეგრირებული ფრეიმვორკის დახმარებით უკავშირდება სისუსტეების ანალიზის სისტემას რათა მოხდეს იმის დადგენა თუ რა სისუსტეები გამოიკვეთა ამა თუ იმ შემთხვევაში. რასაკვირველია თითოეული შემთხვევა შეიძლება იყოს ცალკე განხილვის საგანი მისი უნიკალურობიდან გამომდინარე თუმცა შესაძლოა მოხდეს შემოჭრათა გარკვეული კატეგორიების ჩამოყალიბება , მასში გაერთიანება და შემდეგ გარკვეული დასკვნის გამოტანა და გადაწყვეტილების მიღება.

ასეთი სისტემის მთავარი უპირატესობა გამოიხატება სწავლის მექანიზმში რომელიც ერთ ერთი არსებითი და ძირითადია ასეთ სისტემებში. სწავლის მექანიზმი გულისხმობს შემდეგ მიდგომას: წინასწარ სისტემამ „იცის“ (აქვს ინფორმაცია) გარკვეული შემჭრები, სხვა სიტყვებით რომ ვთქვათ სისტემას შეუძლია გარკვეული ჩარჩოებში მიხვდეს ინფორმაციის ნაკადიდან გამომდინარე შემოჭრაა ეს თუ ჩვეულებრივი შემთხვევა, მაგრამ მთავარი არსი მდგომარეობს იმაში რომ თუ სისტემამ აღმოაჩინა ისეთი ტიპის შემოჭრა რომელიც აქამდე არასოდეს დაფიქსირებულა, მოახდინოს ამ ინფორმაციის დასწავლა და შემდეგ თუ კიდევ განმეორდა ასეთი ტიპის შემოჭრა აღარ დაიწყოს თავიდან ამ ყველაფრის ანალიზი, არამედ მოახდინოს არსებული ცოდნის ბაზიდან ადექვატური დასკვნის გამოტანა თუ რა ტიპის შემოჭრაა ეს. სხვა საკითხია როგორ უნდა მოახდინოს სისტემამ ამ ინფორმაციის შენახვა ცოდნის ბაზებში, როგორი უნდა იყოს ბაზის სტრუქტურა რომ არ დაირღვეს დინამიკა. მთავარი კითხვა ამ შემთხვევაში იქნება როგორი ტიპის უნდა იყოს მონაცემთა ბაზა, რომ მასში მოხდეს ასეთი ტიპის ინფორმაციის შენახვა . ზემოთ განხილული არქიტექტურა რასაკვირველია ეყრდნობა Data Mining ს რომელსაც მოგვიანებით განვიხილავთ და რომელიც გვიპასუხებს ამ შეკითხვაზე.

ერთ ერთი მნიშვნელოვანი საკითხია **სისტემის მოდელირება და ანალიზი** . იგი გულისხმობს გარკვეულ მეთოდოლოგიაზე დაყრდნობით ისეთი მოდულის შექმნას რომელსაც შეეძლება გააანალიზოს არსებული სიტუაცია ხოლო ხოლო თვითონ **ფრეიმვორკი** იყოს გამომძახებელი რომელიც კონკრეტულ სიტუაციაშე ახორციელებს მიმართვას მასზე. სისტემის ანალიზის მოდელი უნდა ეყრდნობოდეს ისეთ ძირითად ნაწილებს როგორებიცაა:

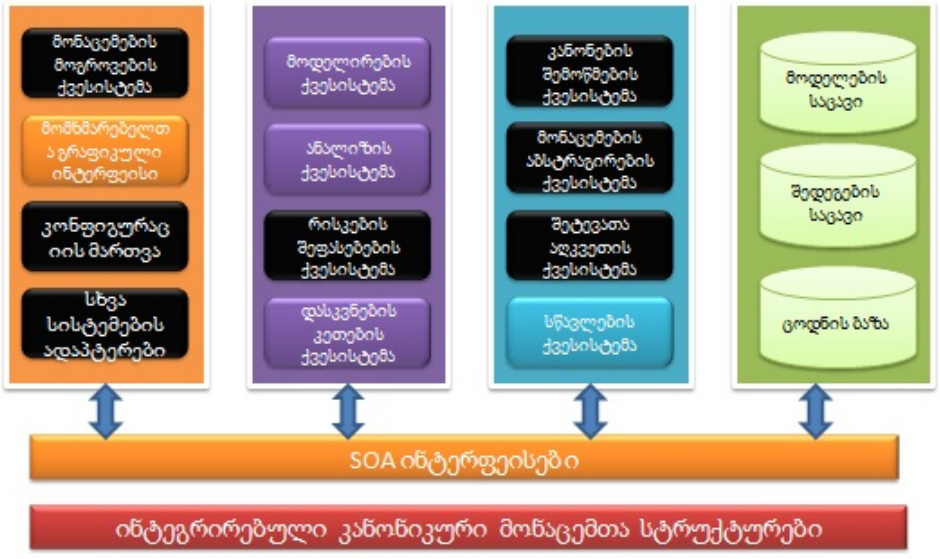
- ალგორითმების დინამიურობა
- თავსებადობა გამომძახებელთან
- ახლის სწავლების უნარი

ალგორითმების დინამიურობა - გულისხმობს ისეთ სისტემას რომლისთვისაც შესაძლებელია ახალი ალგორითმების დამატება/ამოღება საჭიროების შემთხვევაში, ასეთი ტიპის სიტუაციები განპირობებულია ისეთი შემთხვევებით როგორიცაა: ისეთი შემთხვევების აღმოჩენა როცა არსებული ალგორითმები აზრს კარგავს და საჭიროა მათი ჩანაცვლება ახლით ან გაუმჯობესება.

თავსებადობა გამომძახებელთან - გულისხმობს ისეთ არქიტექტურას რომლის თანახმადაც გამომძახებელმა იცის როგორ უნდა გამოიძახოს (შესაძლოა არსებობდეს რაიმე სახის კონტრაქტი რომელშიც აღწერილი იქნება)

ახლის სწავლების უნარი - განხილულ სისტემას უნდა შეეძლოს არსებული შემთხვევის უცოდინრობის შემთხვევაში მისი შესწავლა და დამახსოვრება, იგი უნდა წარმოადგენდეს ინტელექტუალურ აგენტს.

ერთ-ერთ უმთავრეს მოდულს რომელსაც უნდა შეიცავდეს ახალი არქიტექტურის მქონე სისტემა არის უსაფრთხოების მოდული. იგი შეიძლება წარმოვადგინოთ შემდეგი სქემის სახით :



სურ. 66 სისტემის ზოგიერთი ტექნიკური მხარე

როგორც ვხედავთ მთლიანი სისტემა შესაძლოა დაიყოს ქვესისტემებად და გაერთიანდეს სხვადასხვა კატეგორიებში რომელთანაც

უშუალო კავშირი აქვს SOA ინტერფეისებს. სანამ გადავალთ უშუალოდ ამ ქვესისტემების განხილვაზე ჯერ განვმარტოთ რას წარმოადგენს SOA ინტერფეისი.

SOA (Service Oriented Architecture) : Software Engineering ში SOA ეს არის მეთოდოლოგიების გარკვეული პრინციპები რომლებიც საშუალებას იძლევიან მოხდეს გარკვეული წესით ინფორმაციის გაცვლა სხვადასხვა მხარეს შორის. ამ შემთხვევაში სხვადასხვა მხარეში შეიძლება მოისაზრებოდეს ორი ან მეტი ერთმანეთისაგან დამოუკიდებელი სერვისი რომელთაც გარკვეული ოპერაციებისათვის სჭირდებათ სხვა სერვისების გამოყენება, როგორც ზემოთ აღნიშნულ სურათზე ჩანს მას აქვს შესაძლებლობა მიმართოს სხვადასხვა მოდულებს რომლებიც თავის მხრივ არის კლასიფიცირებული შემდეგნაირად:

- <u>I მოდულში გაერთიანებულია შემდეგი ქვემოდულები:</u>
    - **მონაცემების მოგროვების ქვესისტემა** - წარმოადგენს ქვესისტემას რომელიც უზრუნველყობს მონაცემების შეგროვებას თუმცა გამომძახებელს შეიძლება წარმოდგენაც არ ქონდეს იგი რა პრინციპით მუშაობს, გამომძახებელისათვის უბრალოდ მთავარია ამ სისტემისაგან მიიღოს მოთხოვნილი ინფორმაცია, ცალკე საკითხია რა მეთოდოლოგიით მუშაობს მოგროვების ქვესისტემა (შესაძლოა ის იყენებდეს ისეთ ტექნიკას/მიდგომას როგორიცაა მონაცემთა ქეშირება რათა მოხდეს წარმადობის გაზრდა და ა.შ)
    - **მომხმარებელთა გრაფიკული ინტერფეისები** - აღნიშნული მოდული ითვალისწინებს ყველა გრაფიკული ინტერფეისების წარმოდგენას რომელიც შესაძლოა დაჭირდეს ამა თუ იმ ტიპის მომხმარებელს. ეს მოდული უაღრესად მნიშვნელოვანია რადგან მისით ხდება უამრავი სხვა პარამეტრების რედაქტირება,წაშლა, დამატება და ა.შ. იგი მაქსიმალურად უნდა იყოს მორგებული მომხმარებელზე და არ უნდა

საჭიროებდეს რაიმე სპეციფიკურ ცოდნას, წინააღმდეგ შემთხვევაში ასეთი მოდული უბრალოდ გამოუსადეგარი იქნება.

- **კონფიგურაციის მართვა** - სისტემაში შესაძლოა არსებობდეს უამრავი პარამეტრი რასაც იყენებს უსაფრთხოება და რაც ქმნის ერთიანობაში უსაფრთხოების კონფიგურაციას, აქედან გამომდინარე სისტემაში უნდა იქნას დანერგილი კონფიგურაციის მართვის მოდული რათა ყოველ ჯერზე რაიმის ცვლილების გამო არ მოხდეს პროგრამული კოდის გადაკეთება ან თუნდაც ისეთი ადამიანის ჩარევის საჭიროება რომელსაც აუცილებლად გააჩნია რაიმე სპეციფიკური ცოდნა ამ საკითხთან დაკავშირებით. იგი უნდა იყოს მოქნილი და მაქსიმალურად დაცული არასასურველი წვდომისაგან.
- **სხვა სისტემების ადაპტერები** - რაც არ უნდა კარგად იქნას სისტემა დაპროექტებული და რაც არ უნდა რთულ ამოცანებს ასრულებდეს , ნებისმიერ შემთხვევაში შესაძლებელია უბრალოდ ამოცანას დაჭირდეს სხვა სისტემაზე მიმართვა , ამ სხვა სისტემის რაიმე ფუნქციონალობის გამოყენება თავისთან რისი კლასიკური მაგალითიცაა web service ები, რომლის დახმარებითაც სხვადასხვა აპლიკაციის სერვისები შესაძლოა უმარტივესად მიმართავდნენ ერთმანეთს და იყენებდნენ ერთმანეთის ფუნქციონალობას. ყოველივე ზემოთთქმულიდან გამომდინარე სისტემას უნდა გააჩნდეს ასეთი მოდული (უკიდურეს შემთხვევაში უნდა იყოს გათვალისწინებული ასეთი მოდულის არსებობა დაპროექტების ფაზაზე).

- <u>II მოდულში ვაერთიანებთ შემდეგ ქვემოდულებს:</u>
  - **მოდელირების ქვესისტემა** - სისტემაში არსებობს შემავალი მონაცემები, რომლის ნაირსახეობები შესაძლოა იყოს უამრავი და მათი რიცხვის წინასწარ განსაზღვრაც კი არ იყოს

შესაძლებელი, ასეთ სიტუაციაში უნდა მოხდეს კლასიფიცირება და დაჯგუფება მათი რაობის მიხედვით თუმცა სანამ სისტემა ამ ყოველივეს გააკეთებს უნდა არსებობდეს გარკვეული მოდელი, სხვა სისტემით რომ ვთქვათ სისტემა შემავალი მონაცემებიდან გამომდინარე უნდა აგებდეს მოდელს და შემდეგ ახდენდეს ანალიზს თუ რა მოუვა ამა თუ იმ სახის მონაცემს, რატომ შეიძლება იგი საბრთხეს წარმოადგენდეს და ა.შ

- **ანალიზის ქვესისტემა** - წარმოუდგენელია სისტემა რომელიც ამუშავებს სხვადასხვა ტიპის უამრავ ინფორმაციას, ანალიზის სისტემის გარეშე. ამ სისტემამ ძირითადად უნდა უზრუნველჰყოს მონაცემთა სწორი აღქმა რათა შემდგომ შეძლოს სწორი ანალიზის გაკეთება. სხვა საკითხია როგორ უნდა მიეწოდოს ამ სისტემას მონაცემები, ეს კონკრეტულ შემთხვევებზე და კონკრეტულ არქიტექტურაზეა დამოკიდებული თუმცა უკეთეს შემთხვევაში ანალიზის სისტემას არ უნდა უხდებოდეს იმაზე ფიქრი თუ როგორ იქნა მონაცემები მოწოდებული, მან უბრალოდ უნდა მოახდინოს ანალიზი და გამოიმუშაოს ისეთი ინფორმაცია რაც შემდეგ დანარჩენ ორ ქვემოდულს (რისკების შეფასების ქვესისტემას და დასკვნების კეთების ქვესისტემას) დაეხმარება გარკვეული გადაწყვეტილების მიღებაში
- **რისკების შეფასების ქვესისტემა** - არსებობს უამრავი სხვადასხვა ტიპის ინფორმაცია რომელზე დაყრდნობითაც უნდა გაკეთდეს გარკვეული დასკვნები თუმცა თითოეული ტიპი შეიძლება იყოს როგორც მაღალი ასევე დაბალი რისკის მომცველი, ასეთ შემთხვევაში საქმეში ერთვება რისკების შეფასების ქვესისტემა, იგი ისეთნაირად უნდა იყოს დაპროექტებული , რომ შეძლოს გარკვეული

გადაწყვეტილების გამოტანა და შეძლოს ისეთ მთავარ კითხვაზე პასუხის გაცემა როგორიცაა : ამა თუ იმ სიტუაციაში რა რისკს ატარებს მიღებული ინფორმაცია? არის თუ არა ის ბოლომდე სანდო? და ა.შ.

- **დასკვნების კეთების ქვესისტემა** - ყოველივე ზემოთთქმული აზრს დაკარგავდა ამ მოდულის არ არსებობის შემთხვევაში , მას ეკისრება ყველაზე დიდი პასუხისმგებლობა, ამ მოდულმა უნდა შეძლოს და დანარჩენ ქვესისტემებზე დაყრდნობით (მოდელირების ქვესისტემა, ანალიზის ქვესისტემა, რისკების შეფასების ქვესისტემა) შეძლოს გარკვეული დასკვნის გამოტანა. ეს მოდული ყველაზე მნიშვნელოვანია რადგან საბოლოო პასუხი გარკვეულ პრობლემაზე/სიტუაციაზე მასზეა დამოკიდებული, ამ სისტემის პასუხიდან გამომდინარე უნდა მოხდეს გარკვეული ცვლილებების შეტანა და ახალი მოდულების დანერგვა (თუ ამას საჭიროება მოითხოვს).

- <u>III მოდულში ვაერთიანებთ შემდეგ ქვემოდულებს:</u>
  - **კანონების შემოწმების ქვესისტემა** - არსებული სისტემა აუცილებლად უნდა ეყრდნობოდეს გარკვეულ კანონებს, აქ კანონებში იგულისხმება იმ წესების ერთობლიობა რომელიც მოქმედებს გარკვეულ სიტუაციაში გარკვეულ მონაცემებზე, მაგრამ ყველაზე მთავარი დეტალი რაც ამ სისტემას უნდა გააჩნდეს არის კანონების დინამიკა. იგი ეყრდნობა მეთოდოლოგიას რომლის თანახმადაც კანონების შექმნა და რედაქტირება უნდა შეეძლოს იმ ადამიანს ვისაც აქვს არსებულ სისტემაზე წვდომა წინააღმდეგ შემთხვევაში საჭირო გახდებოდა პროდუქტის ყოველ ჯერზე ცვლილება რაც არაეფექტურია და მოითხოვს უდიდეს ძალისხმევას. ამ ქვემოდულმა უნდა შეამოწმოს მიღებული ინფორმაციის სანდოობა სისტემაში არსებოლ კანონებზე დაყრდნობით და

მიიღოს ადექვატური გადაწყვეტილება ნებისმიერ სიტუაციაში.

- **მონაცემების აბსტრაგირების ქვესისტემა** - როგორც უკვე აღინიშნა არსებობს უამრავი სახის ინფორმაცია, მათ შესაძლოა მოეძებნებოდეთ საერთო სახის ატრიბუტები, ხდებოდეს გარკვეული კანონზომიერების დაჭერა ამ ინფორმაციებს შორის, თუმცა შეიძლება გარკვეული ცნებები ეყრდნობოდეს სხვადასხვა განმარტებებს კონკრეტულ სიტუაციაში, ასეთ დროს უნდა მოხდეს მონაცემთა აბსტრაგირება რომელსაც აღნიშნული სისტემა უნდა ახორციელებდეს. სხვა საკითხია მეთოდოლოგია რომლის მიხედვითაც უნდა მოხდეს აბსტრაგირება თუმცა ფაქტია, რომ სწორი არქიტექტურის შემთხვევაში იგი ერთ-ერთ უმთავრეს როლს ასრულებს სისტემაში (როგორც მთლიან წარმადობაში ასევე ცნებების იდენტიფიცირებაში)
- **შეტევათა აღკვეთის ქვესისტემა** - არსებობს დიდი საბრთხე რომელიც მოიცავს სხვადასხვა სახის შეტევებს, ასეთ დროს უკეთეს შემთხვევაში სისტემამ იცის , რომ მიმდინარე ინფორმაცია არის შეტევაზე ორიენტირებული და მიზნად ისახავს სისტემისათვის ზიანის მიყენებას. აქ შეიძლება გაჩნდეს შემდეგი კითხვა : თუ სისტემამ იცის რომ მიმდინარე ინფორმაცია ორიენტირებულია შეტევაზე მაშინ ყოველთვის შეუძლია სისტემას აღკვეთოს იგი? რა თქმა უნდა აღკვეთისათვის ნახევარი საქმეა როცა ამ მოდულისათვის უკვე ცნობილია მიმდინარე ინფორმაციის საბრთხე თუმცა ყველა შემთხვევაში სისტემამ შესაძლოა მაინც ვერ აღკვეთოს იგი. საჭიროა ისეთი მეთოდოლოგიის შემუშავება რაც მაქსიმალურად გაზრდის სისტემის წარმადობას ასეთ შემთხვევაში, სისტემა შეძლებს ინფორმაციის სრულად

შესწავლას და არ გამოეპარება არც ერთი მთავარი დეტალი საბრთხის შესახებ.

- **სწავლების ქვესისტემა** - რაც არ უნდა კარგად და გამართულად მუშაობდეს შეტევათა აღმკვეთი ქვესისტემა იგი მაინც აზრს დაკარგავდა სწავლების ქვესისტემის გარეშე, სწავლების ქვესისტემის დამატებით უბრალოდ პროდუქტი რომელიც კონკრეტულ ჩარჩოებში მუშაობს გადაიქცევა ინტელექტუალურ სისტემად , რომელსაც შეუძლია ისწავლოს და შემდეგ სიტუაციაში აღარ მოუხდეს ყველაფრის თავიდან შესწავლა/ანალიზი (ანალიზის და სხვა მნიშვნელოვანი ფაზების გარეშე სისტემის ადექვატური პასუხი უკვე მიუთითებს მის მაღალ წარმადობაზე რაც ზრდის პროდუქტიულობასაც)

- <u>IV მოდული წინა დანარჩენ სამთან შედარებით არის სპეციფიკური იგი მოიცავს ინფორმაციას ინფორმაციაზე და იყოფა შემდეგ ქვემოდულებად:</u>
  - **მოდელების საცავი** - ჩვენ უკვე განვიხილეთ მოდელის ცნება ახლა უკვე მთავარია ვიცოდეთ როგორ უნდა მოხდეს მისი შენახვა ერთხელ რათა შემდეგ მრავალჯერ გამოვიყენოთ საჭიროების შემთხვევაში, რა თქმა უნდა უნდა არსებობდეს „საცავი“ სადაც სპეციალური ფორმატით შენახული იქნება აღნიშნული მოდელები ეს „საცავი“ არის პირობითი , იგი შეიძლება იყოს უბრალოდ მონაცემთა ბაზა ან ფაილური სისტემა და ა.შ. მთავარი ამ შემთხვევაში არის ის მექანიზმი რომელიც მოდელების საცავს იყენებს. იბადება კითხვა რაში ჭირდება გარკვეულ მექანიზმს მოდელების საცავი, ხომ შეუძლია სისტემას არსებული ინფორმაციის საფუძველზე შეადგინოს მოდელი? აქ არის ერთი ძალიან არსებითი რამ: სისტემა ყოველ ჯერზე არ ახდენს მოდელირებას თუ არსებობს

უკვე მისთვის საჭირო მოდელი იგი მიმართავს და იყენებს მას , ეს ყოველივე რა თქმა უნდა აისახება წარმადობაზე. ასეთ შემთხვევაში სისტემა პირდაპირ O(1) დროში ახორციელებს წვდომას მისთვის საჭირო მოდელზე.

- **შედეგების საცავი** - ასეთი ტიპის ინფორმაციის არსებობა არის განპირობებული შემდეგი გარემოებით: როდესაც სისტემამ უკვე „იცის“ კონკრეტულ სიტუაციაში რა გადაწყვეტილება უნდა იქნას მიღებული მას შეუძლია ეს ინფორმაცია შეინახოს რათა იგივე სიტუაციის გამეორების შემთხვევაში არ მოხდეს მისი თავიდან გააანალიზება, სწორედ ამისათვის უნდა იყოს გამოყენებული შედეგების საცავი. კონკრეტულ სიტუაციაში სისტემა უბრალოდ მოახდენს მასზე მიმართვას და ამოიღებს საჭირო ინფორმაციას.
- **ცოდნის ბაზა** - როგორც უკვე აღვნიშნეთ ჩვენი სისტემა უნდა იყოს ინტელექტუალური სისტემის ერთ - ერთი სახე, მას უნდა ქონდეს ცოდნის გარკვეული ბაზა საწყის ეტაპზე, შემდეგ კი დროდადრო უნდა ხდებოდეს ცოდნის გაზრდა და დახვეწა (ფაქტიურად ისე როგორც ადამიანის შემთხვევაში ხდება). მნიშვნელოვანი არის არა ის თუ ტექნოლოგიურად როგორ მოხდება ამ ბაზის შენახვა არამედ ის თუ როგორი სახით იქნება იგი წარმოდგენილი. რასაკვირველია ცოდნა შესაძლოა უამრავი სახით იყოს შენახული როგორც პირდაპირ მონაცემთა ცხრილებში ასევე ირიბადაც (ფაილებში და ა.შ) თუმცა ყველაზე მთავარია, დაპროექტების ფაზაზე გადაწყდეს სწორი არქიტექტურა აღნიშნული ბაზის რათა ცოდნის გაზრდის და მოდიფიცირების შემდეგ არ გახდეს საჭირო არსებული არქიტექტურის გადაწერისა.

განხილული ქვემოდულები არის ერთ-ერთი არსებითი რაც აღნიშნულ სისტემას უნდა გააჩნდეს თუმცა სპეციფიკაციიდან გამომდინარე შეიძლება დაემატოს კიდევ სხვა მოდულები.

მთავარი განმასხვავებელი ფაქტი ჩვეულებრივ პროდუქტსა და ჩვენს მიერ შემოთავაზებულ სისტემას შორის არის ის, რომ სისტემას შეუძლია ახლის ათვისება, იგი არის დინამიური და ყოველ ახალ პრობლემაზე არ საჭიროებს პროგრამისტის და სხვა კომპეტენტური ადამიანების ჩარევას მისი წარმატებული ფუნქციონირებისათვის.

დასკვნა

ნაშრომში განვიხლეთ ის ზოგადი პრინციპები და მათი თვისებები, რაც ახასიათებს ინტელექტუალურ სისტემას და მანქანურ სწავლებას, რომელიც მისი ერთ ერთი შემადგენელი ნაწილია. დავახასიათეთ სხვადასხვა ალგორითმები გარკვეული მაგალითის საფუძველზე და ხაზი გავუსვით იმ ფაქტს, რომ მხოლოდ ერთი კონკრეტული ალგორითმის გამოყენება არ იძლევა ეფექტურ შედეგს. საბოლოოდ მოვახდინეთ K-Means და გადაწყვეტილებათა ხეების (Decision Tree) კომბინაცია, რომლისგანაც მიიღება საბოლოო მოდელი. მოვიფიქრეთ შესაბამისი არქიტექტურა, რომლის დახმარებითაც შეგვიძლია შევქმნათ შესაბამისი პროგრამული უზრუნველჰყოფა. რასაკვირველია ეს სისტემა არ იქნება მხოლოდ პროგრამა, რომლის საშუალებითაც გადაიჭრება კონკრეტული ამოცანები, არამედ მეტწილად იქნება ინტელექტუალური სისტემა, რომელსაც შეეძლება ახალი ინფორმაციის დამუშავება, დასწავლა და თვითგანვითარება. ამ ყოველივეს ხარჯზე კი აღნიშნული სისტემა მიიღებს და შემოგვთავაზებს გარკვეულ გადაწყვეტილებებს.

აღნიშნული ინტელექტუალური სისტემის განვითარებაზე იქნება დამოკიდებული სხვადასხვა დარგის ამოცანების გადაწყვეტა, რაც უდაოდ ეფექტური იქნება, რადგან აღნიშნულ მოდელზე შექმნილი სისტემა არ იქნება განსაზღვრული მხოლოდ ერთი რომელიმე დარგის ამოცანების გადასაჭრელად, არამედ იგი უნდა მოიცავდეს ერთდროულად ბევრ სხვადასხვა დარგს.

რაც შეეხება განხილულ ალგორითმებს, სრულიად შესაძლებელია არ დავეყრდნოთ მათ კომბინაციას და მოვახდინოთ კიდევ სხვა ალგორითმების გამოყენება (ამ ალგორითმებთან ერთად ან ცალკე განვსაზღვროთ სხვა ალგორითმების მიმდევრობა). აღნიშნული საკითხი წარმოადგენს სამომავლო კვლევის საგანს, რათა დადგინდეს რომელი კომბინაციები არის ყველზე ეფექტური.

გამოყენებული ლიტერატურა

1. https://www.google.com/selfdrivingcar/
2. Artificial Intelligence for Humans Volume 1 Fundamental Algorithms
3. https://www.brainasoft.com/braina/
4. https://en.wikipedia.org/wiki/Mycin
5. https://www.appdynamics.com
6. https://play.google.com/store/apps/details?id=com.netgear.WiFiAnalytics&hl=en
7. http://www.slideshare.net/salahecom/data-mining-concepts-and-techniques-fp-basic
8. Artificial Intelligence. Copyright © 2004 by Massachusetts Institute of Technology.
9. http://www.edureka.co/blog/introduction-to-clustering-in-mahout/
10. http://mnemstudio.org/clustering-k-means-example-1.htm
11. Designing Interfaces 2nd edition by Jenifer Tidwell
12. Server Architectures: Multiprocessors, Clusters, Parallel Systems, Web Servers, Storage Solutions , 1st edition by Rene J. Chevance
13. Database Systems: Design, Implementation, & Management by Carlos Coronel and Steven Morris
14. Enterprise Application Servers CookBook: Part 3: IBM Websphere by Francesco Marchioni
15. Introducing Windows Server 2016 Technical Preview by John McCabe
16. Introduction to Expert Systems, third edition by Peter Jackson
17. SOAP : Cross Platform Web Services Development Using XML by Scott Seely and Kent Sharkey
18. HTTP: The Definitive Guide, 1st edition by David Gourley, Braian Totty, Marjorie Sayer, Anshu Aggarwal, Sailu Reddy
19. The Jboss Group: JBoss Administration and Development by Scott Stark, Marc Fleury

20. Apache Cookbook : Solutions and Examples for apache Administrators 2nd edition by Rich Bowen and Ken Coar
21. SQL : The Ultimate Beginner's Guide by Andrew Johansen
22. Computer Vision : Algorithms and Applications by Richard Szeliski
23. Introduction to Android Application Development: Android Essentials, 5th edition by Joseph Annuzzi jr, Lauren Darcey, Shane Conder
24. IOS Programming : The Big Nerd Ranch Guide , 4th edition by Joe Conwey, Aaron Hillegass, Christian Keur
25. Lean BlackBerry 10 App Development: A Cascades Driven Approach, 2014th edition by Anwar Ludin