საქართველოს ტექნიკური უნივერსიტეტი

ხელნაწერის უფლებით

დავით ჭოხონელიძე

მანქანური სწავლების კომბინირებული ალგორითმების დამუშავება

დოქტორის აკადემიური ხარისხის მოსაპოვებლად წარდგენილი

დისერტაციის

ა ვ ტ ო რ ე ფ ე რ ა ტ ი

სადოქტორო პროგრამა „ინფორმატიკა“ შიფრი 0401

თბილისი
2016 წელი

სამუშაო შესრულებულია საქართველოს ტექნიკური უნივერსიტეტში
ინფორმატიკისა და მართვის სისტემების ფაკულტეტი
ეკონომიკური ინფორმატიკის დეპარტამენტი

ხელმძღვანელი: პროფ. ზურაბ ბოსიკაშვილი

რეცენზენტები: ----------

დაცვა შედგება ---------- წლის ”------” ----------, ---------- 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

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. ინტელექტუალური სისტემები საკმაოდ პოპულარულია და გააჩნიათ კარგი გამოყენება. ინტელექტუალური სისტემა თავის მხრივ მოიცავს ისეთ განუყოფელ დარგებს, როგორიცაა: მანქანური სწავლება, მონაცემთა ანალიზი და შესაბამისად შენახვა (ე.წ Data Mining ი). რაც დრო გადის ინფორმაციის ნაკადი მკვეთრად იზრდება და შესაბამისად ამ ინფორმაციის დასწავლასა და დამუშავებას ინტელექტუალური სისტემა გაცილებით ეფექტურად და ჩქარა შეძლებს ვიდრე ადამიანი. 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.

კვლევის საგანი: დისერტაციის კვლევის საგანს წარმოადგენს მანქანური სწავლების ალგორითმები და მათი კომბინაცია, კონკრეტულ ალგორითმებში შემავალი ინფორმაციის ნორმალიზაციის პრინციპი. მისი გამოყენება მოხდება კონკრეტული არქიტექტურის მქონე ინტელექტუალურ სისტემაში (რომლის არქიტექტურასაც ნაშრომში გნვიხილავთ).

სამეცნიერო ბაზისი: მონაცემთა ნორმალიზაცია/დენორმალიზაცია წარმოადგენს ამომავალ წერტილს ინტელექტუალური სისტემის გამართული მუშაობისათვის. ძალიან მნიშვნელოვანია ხელმძღვანელის მიერ შემუშავებული მთლიანი სისტემის არქიტექტურა (მთლიანი სისტემა

ფრეიმვორკის ჩათვლით), რომელზე დაყრდნობითაც შესაძლოა შეიქმნას ახალი ტიპის ინტელექტუალური სისტემა.

კვლევის მიზანი: კვლევის მიზანს წარმოადგენს ისეთი არქიტექტურის შემუშავება, რომელიც საფუძველს ჩაუყრის ახალი ტიპის ინტელექტუალური სისტემის შექმნას. აღნიშნულ არქიტექტურაში გამოყენებულ იქნება კონკრეტული ალგორითმების კომბინაცია. მთლიანი სისტემის არქიტექტურა მოიცავს უამრავ სხვადასხვა კომპონენტს. კვლევის მიზანს ასევე წარმოადგენს ძირითად ბირთვში წარმოდგენილი ალგორითმები და მათი შემავალი და გამომავალი ინფორმაციის ნორმალიზაცია/დენორმალიზაცია.

მეცნიერული სიახლე: სადისერტაციო ნაშრომში მეცნიერულ სიახლეს წარმოადგენს:

- ნორმალიზაციის პრინციპი, რომელიც შეგვიძლია მოვარგოთ გარკვეული ტიპის ალგორითმებს და მისი საშუალებით მოვახდინოთ საწყისი ინფორმაციის გარდაქმნა;
- სისტემის კომპონენტები, რომელშიც ერთიანდება მონაცემთა დამუშავების, უსაფრთხოების, შემავალი და გამომავლი ინფორმაციის გარდაქმნის მეთოდები და ტექნოლოგიები;

აღწერილი ორი პუნქტის რეალიზაციის გაერთიანება გვაძლევს ახალ მიდგომას, რომელიც დამყარებულია ერთ მთლიან არქიტექტურაზე. აღნიშნული არქიტექტურის საშუალებით მკვეთრად გაუმჯობესდება ინტელექტუალური სისტემების შექმნა და განვითარება სხვადასხვა დარგებისთვის (მედიცინა, ბიოლოგია და ა.შ), გადაიჭრება ისეთი ამოცანები, როგორიცაა: უსაფრთხოების სისტემის ანალიზი, სისუსტეების აღმოჩენა, უსაფრთხოების სისტემაში შემოჭრების აღმოჩენა და ა.შ.

დისერტაციის პრაქტიკული მნიშვნელობა: თანამედროვე დროში ინტელექტუალური სისტემების შექმნა და განვითარება საკმაოდ აქტუალურია. არსებობს უკვე შექმნილი ინტელექტუალური სისტემები, რომლებიც

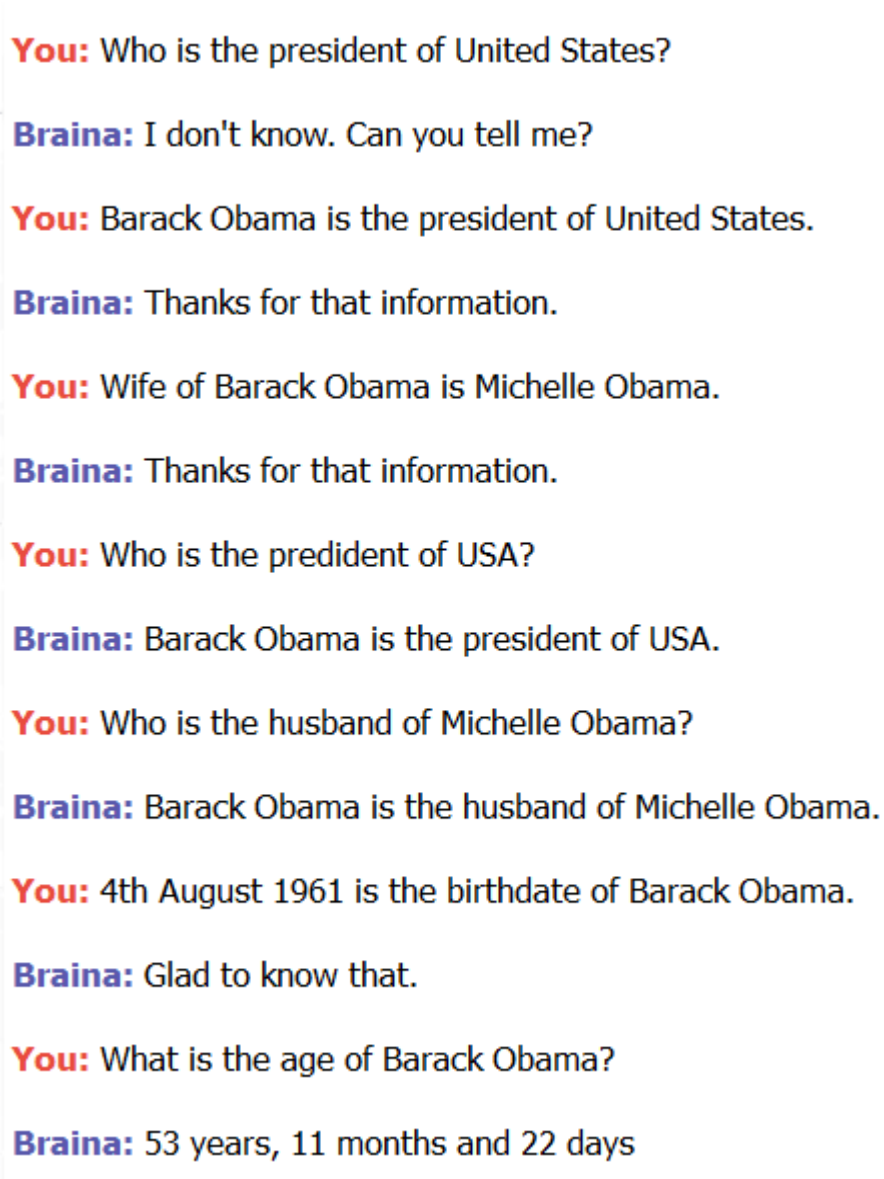
გამოიყენება კონკრეტულ დარგებში. სადისერტაციო ნაშრომში წარმოდგენილი სიახლე გვთავაზობს მიდგომას, რომლის მიხედვითაც შეიძლება შეიქმნას და განვითარდეს ახალი არქიტექტურის ტიპის მქონე ინტელექტუალური სისტემები. მათი დახმარებით გადაიჭრება შემდეგი ტიპის ამოცანები უფრო ეფექტურად: უსაფრთხოების, ბიოლოგიური სისტემის ანალიზი, მათში სისუსტეების აღმოჩენა. საბოლოო ჯამში ინტელექტუალური სისტემა შემოგვთავაზებს გადაწყვეტილებებს კონკრეტულ სიტუაციებში.

დისერტაციის მოცულობა და სტრუქტურა: დისერტაცია შედგება 106 (ასოთხი) გვერდისაგან და მოიცავს შემდეგ საკითხებს: არსებული სისტემების ზოგადი მიმოხილვა, ერთ-ერთი ძირითადი ამოცანა, მეთოდები და ალგორითმები, ექსპერიმენტი კონკრეტულ მონაცემებზე და ბოლოს წარმოადგენს სისტემის არქიტექტურას, როგორც პროგრამულ მოდელს.

თავი 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 , Mycin. 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:

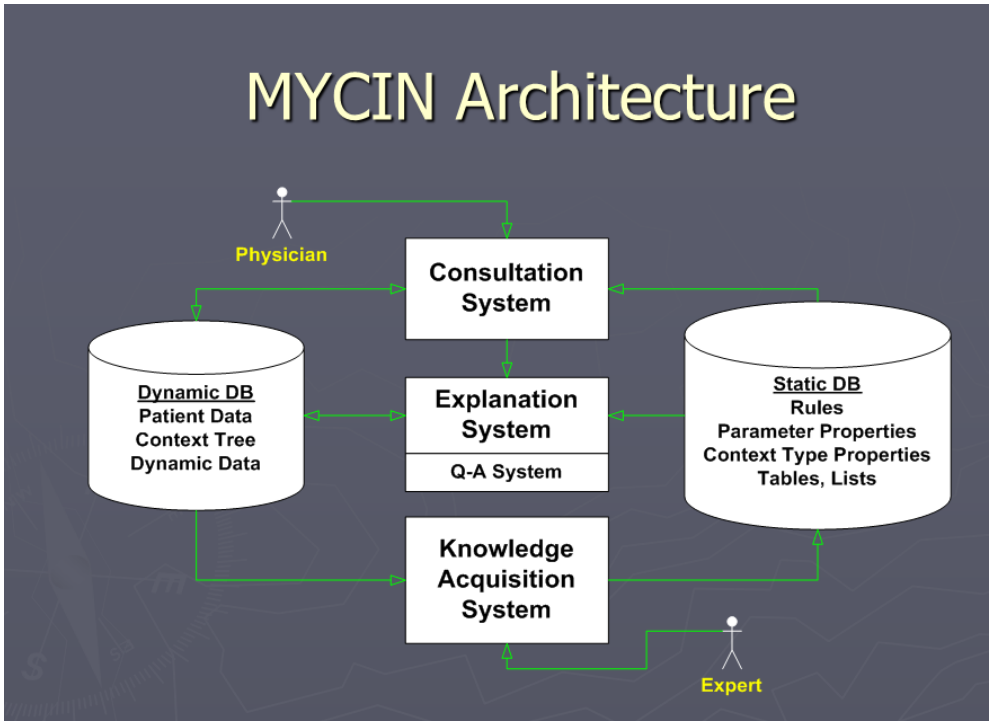


სურ. 1 ინფორმაციის დასწავლა

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“.

Mycin წარმოადგენს ექსპერტულ სისტემას/ინტელექტუალურ სისტემას, რომელიც გამოიყენებოდა 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. 2 Mycin ის არქიტექტურა

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. სანამ უშუალოდ ამ მოდელის შექმანზე ვისაუბრებდეთ ჯერ განვიხილოთ რომელიმე ამოცანა, რომელსაც შეიძლება მოვარგოთ ჩვენს მიერ შემუშავებული არქიტექტურა.

თავი 2: კონკრეტული 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).

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.

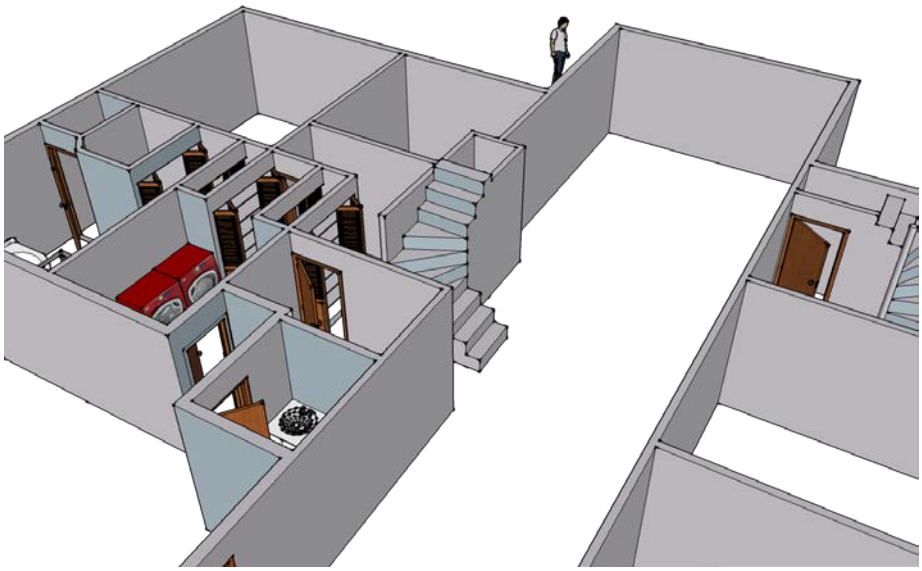
ერთ-ერთი ასეთი ტიპის პროგრამული უზრუნველჰყოფაა App Dynamics. მისი საშუალებით ხდება აპლიკაციის დონეზე უამრავი სხვადასხვა ტიპის მონიტორინგი. მოიცავს სხვადასხვა პროგრამირების ენებს:

სურ. 3 სხვადასხვა დაპრ.ენების და ტექნოლოგიების მხარდაჭერა

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.

მაგალითად შეგვიძლია მოვიყვანოთ ერთი საინტერესო ამოცანა, რომლის გადაწყვეტის გზაც შეგვიძლია ვიპოვოთ ჩვენს მიერ შემუშავებული მოდელის საფუძველზე. 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.

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



სურ. 4 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. 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. ამოცანის გადაწყვეტა მოითხოვს გარკვეული ალგორითმების ამორჩევას. იმისათვის, რომ მოვახდინოთ კონკრეტული ალგორითმების განსაზღვრა ჯერ განვიხილოთ რამოდენიმე სხვადასხვა ტიპის ალგორითმი.

თავი 3: მეთოდები და ალგორითმები

პირველ რიგში განვიხილოთ 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. 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.

კლასიფიკაციის ერთ-ერთ კლასიკურ ალგორითმს წარმოადგენს გადაწყვეტილებათა ხეები (Decision Tree). 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:

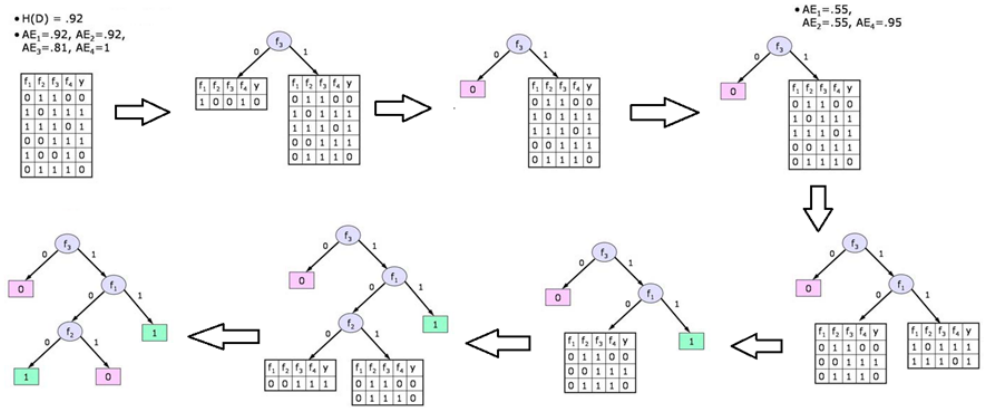
$I_i(p_0,n_0) = - \frac{p_0}{p_{0+}n_0} log_2(\frac{p_0}{p_{0+}n_0}) - (1 - \frac{p_0}{p_{0+}n_0}) log_2(1 - \frac{p_0}{p_{0+}n_0})$ და

$I_i(p_1,n_1) = - \frac{p_1}{p_{1+}n_1} log_2(\frac{p_1}{p_{1+}n_1}) - (1 - \frac{p_1}{p_{1+}n_1}) log_2(1 - \frac{p_1}{p_{1+}n_1})$;

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:

მთლიანი პროცესი შეგვიძლია შემდეგნაურად წარმოვადგინოთ:



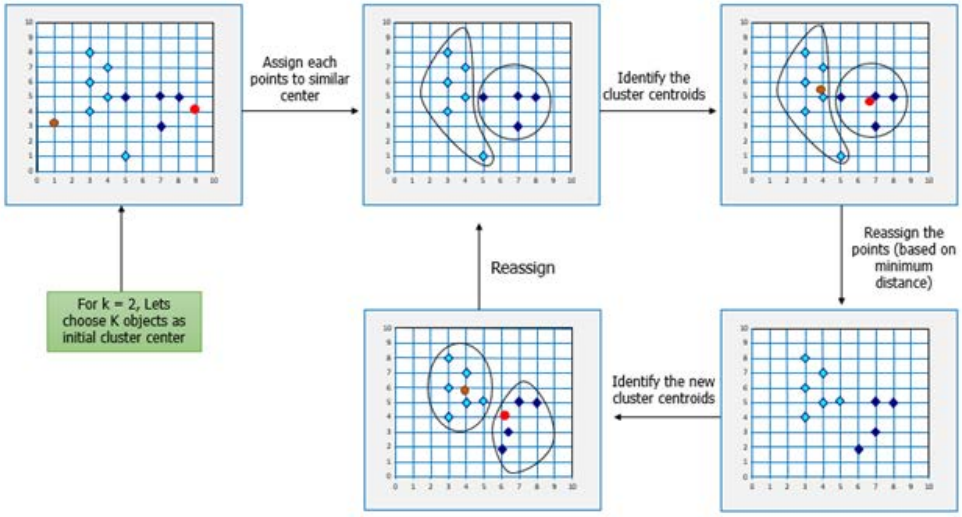
სურ. 5 გადაწყვეტილებათა ხე, აგების მთლიანი პროცესი

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;

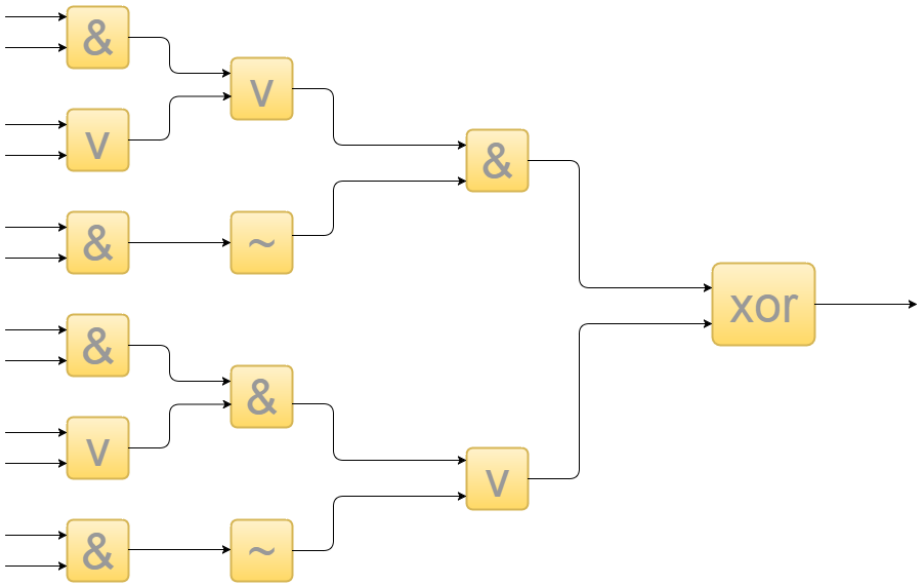
წერტილების გადანაწილების შემდეგ (წინა პუნქტის შედეგად ხდება გარკვეული წერტილების სხვა კლასტერზე მინიჭება, რადგან მის ცენტროიდთან უფრო ახლოს არის ეს წერტილი) ვიპოვოთ ცენტროიდი ახალ კლასტერში.



სურ. 6 K-Means ალგორითმის სიმულაცია

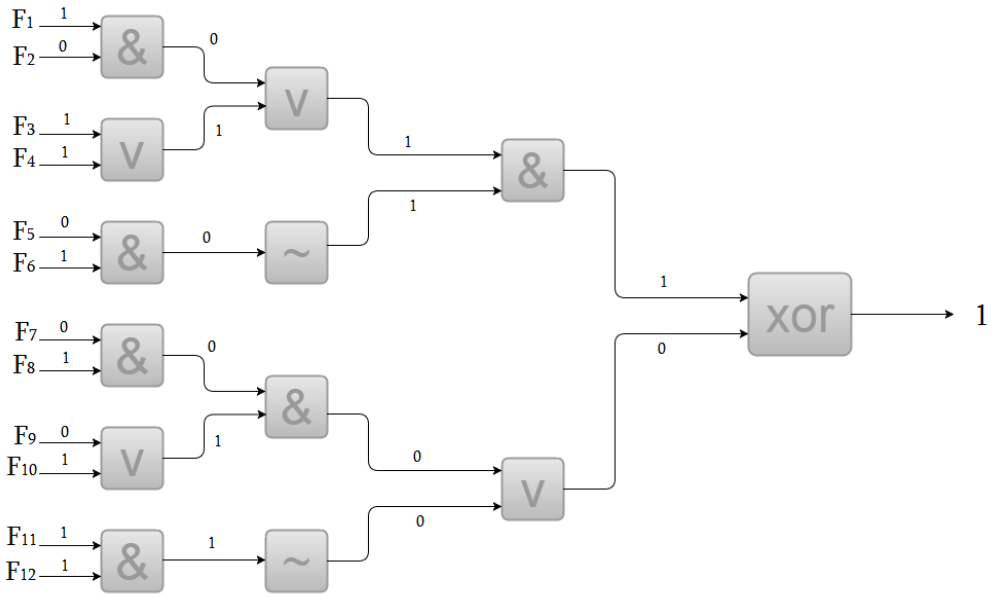
სისტემაში ინფორმაცია დავყავით ორ ძირითად კატეგორიად, რომელთაგანაც ერთ ერთს წარმოადგენს შემავალი ინფორმაცია. სანამ უშუალოდ სისტემა დაიწყებს თავისი ალგორითმების გამოყენებას, აუცილებელია მოხდეს გარდაქმნა მონაცემების (ნორმალიზაცია). ნებისმიერი ალგორითმისათვის აუცილებელია შემავალი ინფორმაცია. როგორც უკვე აღვნიშნეთ გადაწყვეტილებათა ხეების 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 სქემა, რომელსაც შეგვიძლია ვუწოდოთ ლოგიკური სქემა, რომელიც განსაზღვრულია 12 ოპერანდზე:



სურ. 7 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. 8 ლოგიკური სქემის გამოყენების მაგალითი

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

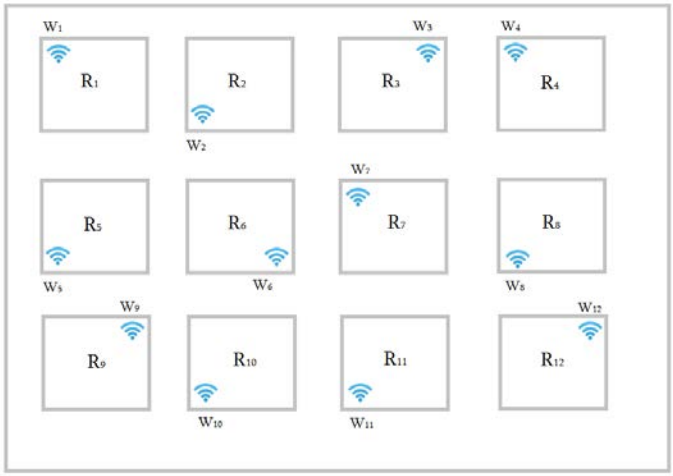
სურ. 9 კლასიფიკაციისა და კლასტერიზაციის შედარება

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. იმისათვის, რომ ალგორითმების ამორჩევა უფრო ეფექტური იყოს, შეგვიძლია ჩავატაროთ ექსპერიმენტი კონკრეტულ მონაცემებზე.

თავი 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. 10 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;

მხოლოდ ერთი ალგორითმის გამოყენება არ არის ეფექტური, ამიტომ მოვახდინოთ რამოდენიმე ალგორითმის კომბინაცია.

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:

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

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

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

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

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

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

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

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

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

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

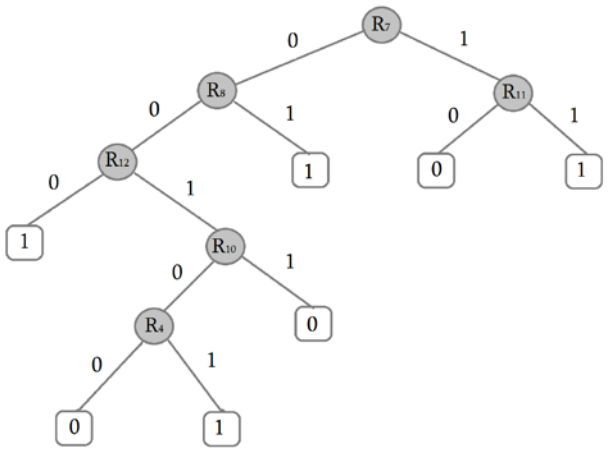
$$R_{11} = \{1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0\};$$

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

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

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:



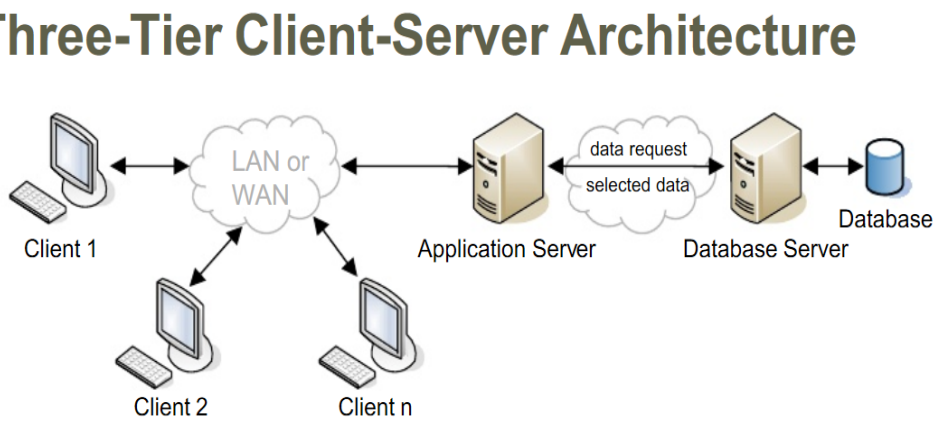
სურ. 11 ექსპერიმენტის შედეგად მიღებული გადაწყვეტილებათა ხე

ექსპერიმენტული ნაწილი წარმოადგენს კარგ საშუალებას რათა ვაჩვენოთ თუ როგორ მუშაობს კონკრეტული ალგორითმები. არანაკლებ მნიშვნელოვანია გამოყენებული ალგორითმების რეალიზაცია პროგრამული სახით და შესაბამისი არქიტექტურის წარმოდგენა.

თავი 5: მთლიანი პროცესის პროგრამული არქიტექტურის სახით წარმოდგენა

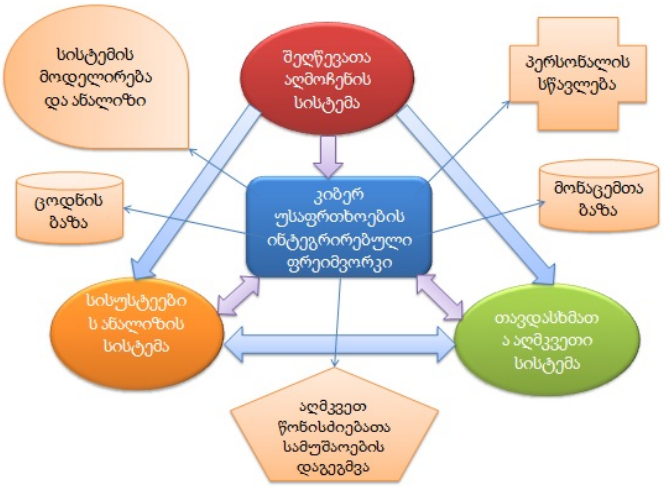
ინტელექტუალური სისტემა შესაძლოა განხილულ იქნას როგორც პროგრამა, რომელიც სხვა ჩვეულებრივი პროგრამებისაგან განსხვავებით გამოირჩევა დასწავლის უნარით და აზროვნებს გარკვეულ დონეზე, სწავლობს გამოცდილებიდან და აქვს უნარი გააუმჯობესოს საკუთარი თავი. სანამ უშუალოდ ჩვენს მიერ შეთავაზებულ არქიტექტურაზე გადავალთ, 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).

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.



სურ. 12 სამ დონიანი არქიტექტურა

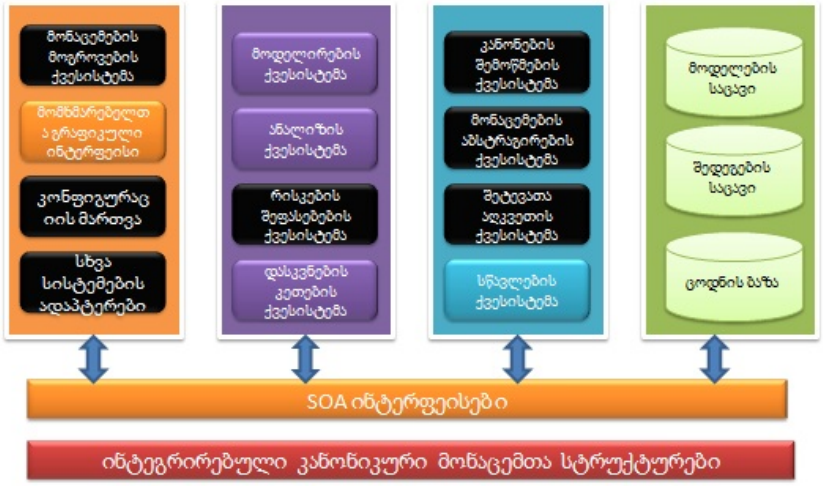
ჩვენს მიერ შემუშავებული არქიტექტურა მოიცავს ისეთ ძირითად ნაწილებს როგორიცაა კიბერ უსაფრთხოების ინტეგრირებული ფრეიმვორკი, ცოდნის ბაზა და ა.შ. ეს ყველაფერი შეგვიძლია წარმოვადგინოთ შემდეგი სურათის სახით:



სურ. 13 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.

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



სურ. 14 სისტემის ზოგიერთი ტექნიკური მხარე

როგორც ვხედავთ მთლიანი სისტემა შესაძლოა დაიყოს ქვესისტემებად და გაერთიანდეს სხვადასხვა კატეგორიებში რომელთანაც უშუალო კავშირი აქვს SOA ინტერფეისებს.

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.

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.

- <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.

დასკვნა

ნაშრომში განვიხილეთ კონკრეტული ალგორითმები და ახალი ტიპის არქიტექტურა, რომლის საშუალებითაც შეიძლება შეიქმნას ინტელექტუალურ სისტემა. განხილული ალგორითმები გამოყენებულ იქნება მანქანურ სწავლებაში, რომელიც ინტელექტუალური სისტემის ერთ-ერთი შემადგენელი ნაწილია. მოვახდინეთ სხვადასხვა ალგორითმების რეალიზაცია კონკრეტულ მაგალითებზე 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.

შემუშავებული ინტელექტუალური სისტემის არქიტექტურის გამოყენებასა და 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.

დისერტაციის თემასთან დაკავშირებით გამოქვეყნებული პუბლიკაციები

1. დ.ჭოხონელიძე, ზ.ბოსიკაშვილი „მანქანური სწავლების კლასიფიკაციის ალგორითმებში შემავალი ინფორმაციის ფორმირება“, მართვის ავტომატიზირებული სისტემები, N2 (20), თბილისი, 2015, გვ 31-35;
2. დ.ჭოხონელიძე, ზ.ბოსიკაშვილი „მანქანური სწავლების კლასტერიზაციის ალგორითმებში გამომავალი ინფორმაციის ფორმირება“, მართვის ავტომატიზირებული სისტემები, N2 (20), თბილისი, 2015, გვ 36-41;
3. დ.ჭოხონელიძე, ზ.ბოსიკაშვილი „მანქანური სწავლების ალგორითმებში შემავალი და გამომავალი ინფორმაციის ნორმალიზაცია“, მართვის ავტომატიზირებული სისტემები, N1 (21), თბილისი, 2016, გვ 133-137;
4. დ.ჭოხონელიძე, ზ.ბოსიკაშვილი „მანქანური სწავლების კომბინირებული ალგორითმები“, აკადემიკოს ი. ფრანგიშვილის დაბადების 85-ე წლისთავისადმი მიძღვნილი საერთაშორისო სამეცნიერო კონფერენცია «საინფორმაციო და კომპიუტერული ტექნოლოგიები, მოდელირება, მართვა» თბილისი, 2015;

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 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.