

UNIVERZITET SINGIDUNUM

Departman za posleddiplomske studije i međunarodnu saradnju

– DOKTORSKA DISERTACIJA –

**SOFTVERSKI SIMULATOR ZA EDUKACIJU U
OBLASTI INTELIGENTNIH SISTEMA**

Mentor: prof. dr Mladen Veinović

Student: Marko Marković, M.Sc.

Broj indeksa: 460055/2011

Beograd, 2014.

Apstrakt

Zbog rastuće važnosti oblasti inteligentnih sistema, izražena je sve veća potreba za izučavanjem algoritama koji se koriste u ovoj oblasti, a koji se oslanjaju na mašinsko učenje i algoritme pretraživanja. Ovakvi algoritmi često mogu biti komplikovani za teorijsko objašnjavanje i samim tim za razumevanje. Kao kvalitetan alat za prevazilaženje problema u njihovom objašnjavanju pokazalo se korišćenje vizuelnih softverskih simulatora. Zbog toga se ukazala potreba da se u ovom radu simulacije predstave kao jedno od najmodernijih rešenja u obrazovanju uz prikaz načina na koje se mogu primeniti u nastavi.

U ovom radu predstavljen je opis i tehnička realizacija softverskog simulatora razvijenog da pomogne izučavanje oblasti inteligentnih sistema na Poslovnom fakultetu Valjevo Univerziteta Singidunum. U okviru algoritama mašinskog učenja obuhvaćena su stabla odlučivanja, klasterovanje, Naive Bayes i perceptroni, a u okviru algoritama pretraživanja obrađeni su pretraživanje po širini i dubini, planinarenje i A*. Na kraju rada, data je evaluacija razvijenog simulatora.

Ključne reči

Softverski simulatori, inteligentni sistemi, mašinsko učenje, algoritmi pretraživanja

SADRŽAJ

1. UVOD	11
1.1. Predmet istraživanja	12
1.2. Ciljevi i zadaci istraživanja.....	12
1.3. Hipotetički okvir istraživanja	13
1.4. Metode i tok istraživačkog postupka	14
1.5. Struktura rada	15
1.6. Doprinos rada.....	16
 2. PREGLED OBLASTI ISTRAŽIVANJA	 18
2.1. Softverske simulacije u savremenoj edukaciji	18
<i>2.1.1. Istorijat softverskih simulacija.....</i>	<i>18</i>
<i>2.1.2. Korisnost softverskih simulacija.....</i>	<i>19</i>
<i>2.1.3. Značaj vizuelnih softverskih simulacija</i>	<i>21</i>
2.2. Inteligentni sistemi u poslovnom odlučivanju	23
<i>2.2.1. Proces donošenja odluka.....</i>	<i>23</i>
<i>2.2.2. Uloga informacionih tehnologija u</i> <i>savremenom poslovnom odlučivanju.....</i>	<i>25</i>
<i>2.2.3. Sistemi za podršku odlučivanju.....</i>	<i>26</i>
2.3. Postojeća rešenja u edukaciji poslovnog odlučivanja	27
<i>2.3.1. TopSim Easy Management.....</i>	<i>28</i>
<i>2.3.2. Harvard Working Capital Simulation</i>	<i>29</i>
<i>2.3.3. Cesim GlobalChallenge.....</i>	<i>30</i>
<i>2.3.4. Forio Simulate</i>	<i>31</i>
<i>2.3.5. SmartSims MikesBikes-Advanced.....</i>	<i>32</i>
<i>2.3.6. JA Titan</i>	<i>33</i>
2.4. Definisanje problema i predlozi za poboljšanja korišćenjem inteligentnih sistema.....	34

3. TEORIJSKE OSNOVE ISTRAŽIVANJA	36
3.1. Faktori upotrebljivosti određenog softvera	36
3.2. Uporedna iskustva i rešenja u korišćenju simulatora inteligentnih sistema	37
3.2.1. Postojeći simulatori algoritama mašinskog učenja	40
3.2.2. Postojeći simulatori algoritama pretraživanja	46
3.3. Analiza izabranih algoritama mašinskog učenja	50
3.3.1. Stabla odlučivanja	50
3.3.2. Klasterovanje	55
3.3.3. Naive Bayes	58
3.3.4. Perceptroni	60
3.4. Analiza izabranih algoritama pretraživanja	64
3.4.1. Pretraživanje po širini	65
3.4.2. Pretraživanje po dubini	67
3.4.3. Planinarenje	69
3.4.4. A*	72
3.5. Predlog rešenja softverskog sistema	75
4. REALIZACIJA NOVOG SOFTVERSKOG SIMULATORA	77
4.1. Prikaz funkcionalnosti sistema	77
4.2. Tehnologije korišćene za izradu sistema	81
4.2.1. Java AWT	82
4.2.2. Java Swing	83
4.2.3. Biblioteka JUNG	84
4.3. Opis funkcionalnosti sistema	85
4.3.1. Opis funkcionalnosti stabala odlučivanja	86
4.3.2. Opis funkcionalnosti algoritma klasterovanje	89
4.3.3. Opis funkcionalnosti algoritma Naive Bayes	92

4.3.4. Opis funkcionalnosti perceptrona.....	95
4.3.5. Opis funkcionalnosti modula za algoritme pretraživanja.....	98
4.4. Implementacija sistema	104
4.4.1. Glavni prozor programa	108
4.4.2. Implementacija stabla odlučivanja	109
4.4.3. Implementacija klasterovanja.....	112
4.4.4. Implementacija algoritma Naive Bayes	114
4.4.5. Implementacija perceptrona.....	116
4.4.6. Implementacija algoritama pretraživanja	119
5. LABORATORIJSKE VEŽBE ZASNOVANE NA SOFTVERSKOM SISTEMU.....	124
5.1. Demonstracija rada stabala odlučivanja	124
5.2. Demonstracija rada klasterovanja.....	128
5.3. Demonstracija rada Naive Bayes-a	132
5.4. Demonstracija rada perceptrona	133
5.5. Demonstracija rada pretrage po širini	136
5.6. Demonstracija rada pretrage po dubini	138
5.7. Demonstracija rada planinarenja	139
5.8. Demonstracija rada algoritma A*	140
6. EVALUACIJA REALIZOVANOG SISTEMA.....	142
6.1. Uzorak podataka	142
6.2. Analiza podataka.....	143
6.3. Diskusija	146
7. ZAKLJUČAK	147
LITERATURA.....	150

Spisak grafikona

Grafikon 1: Uloga sistema za podršku odlučivanju	27
Grafikon 2: Početna strana simulacije TopSim Easy Management.....	28
Grafikon 3: Prikaz toka novca u simulaciji Working Capital Simulation	29
Grafikon 4: Početna strana simulacije Cesim GlobalChallenge.....	30
Grafikon 5: Izgled modela simulacije u okviru Forio Simulate	31
Grafikon 6: Izgled ekrana u simulaciji SmartSims MikesBikes-Advanced	32
Grafikon 7: Izgled jednog od ekrana simulacije JA Titan.....	33
Grafikon 8: Simulator stabla odlučivanja - Univerzitet Novog Južnog Velsa.	40
Grafikon 9: Simulator stabla pretraživanja - Imperijal Koledž.	41
Grafikon 10: Simulator klasterovanja - Tehnički univerzitet Braunšvajg.	41
Grafikon 11: Simulator klasterovanja - Univerzitet Alberta.	42
Grafikon 12: Simulator klasifikacionih algoritama - Tehnion.....	43
Grafikon 13: Simulator perceptrona - Blisko-istočni tehnički univerzitet.....	43
Grafikon 14: Pojednostavljeni simulator fudbala – Univerzitet Delft.....	44
Grafikon 15: Simulator WEKA.....	45
Grafikon 16: Simulator SIMBA.....	45
Grafikon 17: MIT Open CourseWare simulator.	46
Grafikon 18: AI Space simulator.	47
Grafikon 19: Aima3e.....	48
Grafikon 20: AI-Search.....	49
Grafikon 21: Data Structure Visualizations.	49
Grafikon 22: Prikaz stabla odlučivanja.	51
Grafikon 23: Kriva obučavanja za stablo odlučivanja.	53
Grafikon 24: Veza između visine stabla i preciznosti odgovora.	54
Grafikon 25: Prikaz nekoliko načina klasterovanja.	56
Grafikon 26: Demonstracija rada algoritma k-means.....	57
Grafikon 27: Kriva obučavanja za algoritam Naive Bayes. Kao poređenje, prikazana je i kriva obučavanja za stablo odlučivanja na istom primeru.	60
Grafikon 28: Delovi nervne ćelije neurona.	61
Grafikon 29: Podela ulaznog skupa pomoću perceptrona.	62
Grafikon 30: Matematički model neurona.	63
Grafikon 31: Kriva obučavanja za perceptron. Prikazano je poređenje sa stablom odlučivanja u dve specifične situacije.	63
Grafikon 32: Pretraga stabla pomoću BFS algoritma.	65
Grafikon 33: Redosled obilaska stabla pomoću BFS algoritma.....	66
Grafikon 34: Redosled obilaska stabla pomoću DFS algoritma.....	68

Grafikon 35: Obilazak stabla na preorder način.	68
Grafikon 36: Jednostavan primer n-kraljica.	70
Grafikon 37: Problemi sa lokalnim maksimumima u planinarenju.	71
Grafikon 38: Prikaz rada algoritma A* na primeru mape rumunskih gradova.	73
Grafikon 39: Početni ekran aplikacije bez ijednog učitanoog modula.	85
Grafikon 40: Izgled modula stablo odlučivanja.	86
Grafikon 41: Izgled modula nakon unosa podataka.	87
Grafikon 42: Prikaz generisanog stabla odlučivanja.	87
Grafikon 43: Izgled stabla odlučivanja.	88
Grafikon 44: Prikaz informacionih dobiti u kontekstnom meniju.	88
Grafikon 45: Izgled modula klasterovanje.	89
Grafikon 46: Podešavanje početnih vrednosti.	90
Grafikon 47: Prikaz istorije centroida.	90
Grafikon 48: Prikaz tekstualnog izlaza algoritma.	91
Grafikon 49: Linija alata za modul klasterovanje.	91
Grafikon 50: Izgled modula Naive Bayes.	92
Grafikon 51: Izgled modula Naive Bayes nakon učitavanja početnih podataka.	92
Grafikon 52: Izbor željenog test primera.	93
Grafikon 53: Izgled stabla sa uslovnim verovatnoćama.	93
Grafikon 54: Označavanje redova u tabeli sa ulaznim podacima.	94
Grafikon 55: Prikaz izlaza algoritma Naive Bayes.	94
Grafikon 56: Izgled modula perceptron.	95
Grafikon 57: Podešavanje ulaznih parametara algoritma.	95
Grafikon 58: Izbor početnog skupa vrednosti.	96
Grafikon 59: Izgled modula nakon pokretanja algoritma.	96
Grafikon 60: Označavanje reda koji se trenutno koristi.	97
Grafikon 61: Grafički prikaz perceptrona.	97
Grafikon 62: Tekstualni izlaz algoritma.	98
Grafikon 63: Izbor algoritama pretraživanja.	99
Grafikon 64: Početni izgled modula za algoritme pretraživanja.	99
Grafikon 65: Linija sa alatima u modulu Pretraživanje.	100
Grafikon 66: Okvir za pravljenje novog grafa.	100
Grafikon 67: Izgled okvira za izmenu postojećeg grafa.	101
Grafikon 68: Padajuća lista sa opcijama za izmene grafa.	101
Grafikon 69: Opcija "Biranje" za uređivanje grafa.	102
Grafikon 70: Prikaz napravljenog grafa u simulatoru.	102
Grafikon 71: Izgled stabla pretraživanja.	103
Grafikon 72: Dijagram klasa softverskog simulatora.	105
Grafikon 73: Stablo odlučivanja za primer play tennis.	125

Grafikon 74: Informacione dobite za koreni čvor - outlook.....	126
Grafikon 75: Stablo odlučivanja za primer kupovine proizvoda ili usluga.	127
Grafikon 76: Početni prikaz tačaka za tri klastera.	129
Grafikon 77: Prikaz tačaka i centroida na kraju rada algoritma.....	129
Grafikon 78: Početni prikaz tačaka za deset klastera.	130
Grafikon 79: Prikaz tačaka i centroida na kraju rada algoritma.....	131
Grafikon 80: Izbor vrednosti test atributa.	132
Grafikon 81: Prikaz stabla sa vrednostima uslovnih verovatnoća.	132
Grafikon 82: Ulazni podaci za perceptron.	133
Grafikon 83: Izlaz perceptrona za prvi korak algoritma.....	134
Grafikon 84: Početni primer za demonstriranje rada pretrage u širinu.	136
Grafikon 85: Aktivirane opcije za kretanje kroz algoritam.....	136
Grafikon 86: Stablo pretrage nakon prvog koraka pretrage u širinu.	137
Grafikon 87: Stablo pretrage nakon drugog koraka pretrage u širinu.....	137
Grafikon 88: Početni prikaz i trenutno aktivna putanja.	137
Grafikon 89: Krajnji prikaz stabla pretraživanja.....	138
Grafikon 90: Krajnji rezultat pretrage po dubini.	138
Grafikon 91: Primer za demonstraciju rada algoritma planinarenje.	139
Grafikon 92: Simulacija rada algoritma planinarenje.	139
Grafikon 93: Primer za demonstraciju rada algoritma A*.	140
Grafikon 94: Simulacija rada algoritma A*.	140
Grafikon 95: Krajnje rešenje u radu algoritma A*.....	141
Grafikon 96: Prosečna ocena na predmetu Inteligentni informacioni sistemi.	144
Grafikon 97: Procenat studenata koji su položili ispit.	144
Grafikon 98: Broj studenata koji su izabrali izborni predmet.....	145

Spisak listinga programskog koda

<i>Listing 1: Najvažnija polja klase GlavniProzor.java.</i>	108
<i>Listing 2: Osnovna struktura klase TreeNode.</i>	109
<i>Listing 3: Osnovna struktura klase Stablo.</i>	110
<i>Listing 4: Klasa TreeNodePredicate.</i>	112
<i>Listing 5: Prikaz klase Korak.</i>	112
<i>Listing 6: Deo programske logike za iscrtavanje tačaka i klastera.</i>	113
<i>Listing 7: Prikaz padajućih listi za izbor test primera.</i>	114
<i>Listing 8: Prikaz klase ModelTabele.</i>	115
<i>Listing 9: Deo strukture klase CvorNB.</i>	115
<i>Listing 10: Deo metoda pokreniAlgoritam().</i>	116
<i>Listing 11: Programski kod za korigovanje težina.</i>	117
<i>Listing 12: Prilagođavanje grafičke predstave perceptrona.</i>	118
<i>Listing 13: Prilagođavanje grafičke predstave perceptrona.</i>	118
<i>Listing 14: Deo konstruktora klase GraphEditor.java.</i>	119
<i>Listing 15: Osnova klase GrafCvor.</i>	121
<i>Listing 16: Osnova klase GrafVeza.</i>	121
<i>Listing 17: Određivanje boje određenog čvora.</i>	121
<i>Listing 18: Upotreba GridBagLayout menadžera za raspoređivanje komponenti interfejsa.</i>	122
<i>Listing 19: Dodatna klasa GBC.java za rad sa GridBagLayout menadžerom.</i>	123
<i>Listing 20: Tekstualni izlaz algoritma.</i>	131
<i>Listing 21: Tekstualni izlaz algoritma Naive Bayes.</i>	133
<i>Listing 22: Tekstualni izlaz primera za logičku funkciju AND.</i>	135

Spisak tabela

<i>Tabela 1: Ulazni skup podataka primera play tennis.</i>	<i>124</i>
<i>Tabela 2: Ulazni skup podataka za primer kupovine određenog proizvoda ili usluge.</i>	<i>127</i>
<i>Tabela 3: Rezultati istraživanja.</i>	<i>143</i>

1. UVOD

Oblast softverskih simulacija zauzima sve važnije mesto u modernom obrazovanju. Ovakvi softveri pogotovo mogu biti pogodni kada je potrebno razumeti složenije algoritme koji mogu biti komplikovani za objašnjavanje. Tada se ovakvi vizuelni alati mogu pokazati kao velika pomoć s obzirom da se, zahvaljujući njima, i neki složeniji koncepti mogu jednostavnije objasniti i razumeti. To se posebno tačnim može pokazati u oblasti inteligentnih sistema, u kojoj postoji veliki broj algoritama koji su u velikoj meri apstraktni, i u čijem izučavanju se ovakvi vizuelni alati nameću kao izuzetna pomoć i kao novi aspekt metodike nastave. Ovaj pristup je sve popularniji jer omogućuje korisnicima potpuno aktivan pristup učenju u okviru kontrolisanog i bezbednog okruženja.

Savremeno poslovanje se ne može ni zamisliti bez procesa poslovnog odlučivanja, a primena savremenih tehnologija se u tu svrhu nameće kao obavezan alat. S obzirom na to da je ova oblast zbog velike ekspanzije sve interesantnija za proučavanje, posvećuje joj se i sve više pažnje u edukativnom smislu. Korišćenjem softverskih simulacija moguće je koristiti modele situacija iz realnog sveta uz strogo definisane uloge svih učesnika u okruženju bogatom parametrima koji pomažu u donošenju poslovnih odluka. Na ovaj način je moguće steći praktičan uvid u način funkcionisanja poslovnih procesa u modernom preduzeću. Praktično, učesnici imaju mogućnost da upravljaju virtuelnim preduzećem i da utiču na veliki broj parametara donošenjem odgovarajućih odluka. U ovom radu, dat je i prikaz simulatora poslovanja preduzeća.

Analizirani simulatori, iako veoma složeni, ne predstavljaju u potpunosti pogodno rešenje jer ne poseduju mogućnost potpunog objašnjavanja procesa donošenja odluka. U svakom od njih postoji samo mogućnost dobijanja detaljnih izveštaja i bilansa na osnovu kojih je potrebno doneti odluke o ulaganjima, prodaji, cenama proizvoda itd. Problem se nalazi između ove dve faze - korišćenja izveštaja i donošenja odluka. Ne postoji bilo kakva mogućnost da korisnici detaljnije uđu u ovaj proces i dobiju pomoć u analiziranju mnoštva

raznih parametara. Praktično, nakon proučavanja parametara, potrebno je samostalno doneti odluke, a da pritom ne postoji način da se ceo taj proces učini lakšim za razumevanje.

Na ovom polju kao kvalitetno rešenje nameću se inteligentni sistemi. Zahvaljujući algoritmima koji se koriste u ovoj oblasti, mogu se prevazići navedeni problemi, što bi svakako poboljšalo razumevanje oblasti poslovnog odlučivanja. Nijedan od navedenih simulatora ne koristi ovakav pristup, što bi njihovu upotrebljivost svakako moglo da digno na znatno viši nivo.

Tema ovog rada je razvoj jednog takvog simulatora koji je zasnovan na korišćenju algoritama iz oblasti inteligentnih sistema. Za potrebe simulatora, uzete su dve klase inteligentnih algoritama - algoritmi *mašinskog učenja* i *algoritmi pretraživanja*.

Algoritmi mašinskog učenja predstavljaju skup metoda pomoću kojih računari treba da podrže rešavanje problema putem analiziranja (učenja) slučajeva koji su se već dogodili. Algoritmi pretraživanja omogućavaju da se u skupu rezultata pronađe željeni podatak na osnovu određenog identifikatora. U cilju lakšeg razumevanja, sve ove algoritme je moguće predstavljati i grafički, kako bi korisnici imali pregled aktivnosti koje se dešavaju.

1.1. Predmet istraživanja

Predmet istraživanja ovog rada predstavljaju vizuelni softverski simulatori kao edukativni alati koji u poslednje vreme dobijaju sve veću popularnost. Obrađuju se prednosti i mane njihovog korišćenja, značaj koji imaju u edukaciji kao i uporedna analiza postojećih rešenja u praksi - dat je prikaz simulatora poslovnog odlučivanja, mašinskog učenja i algoritama pretraživanja. Na osnovu uočenih nedostataka, predložen je i realizovan novi vizuelni simulator koji se može koristiti za učenje navedenih algoritama.

1.2. Ciljevi i zadaci istraživanja

Opšti cilj ovog rada je objašnjavanje važnosti uloge vizuelnih softverskih simulacija u edukaciji, s obzirom da se često javlja potreba za objašnjavanjem određenih koncepata koji

mogu biti komplikovani za teorijsko prikazivanje, što se posebno može odnositi na algoritme u inteligentnim sistemima.

Posebni ciljevi ovog rada su:

- Ukazivanje na važnost primene inteligentnih sistema u oblasti poslovnog odlučivanja koje predstavlja važan element savremene poslovne ekonomije. Posebno je važno prikazati značaj primene informacionih tehnologija u ovoj oblasti, zahvaljujući čemu se njena upotrebljivost diže na znatno viši nivo.
- Prikazivanje uporedne analize iskustava i rešenja koja već postoje u praksi. Na ovaj način, omogućeno je sagledavanje uloge oblasti softverskih simulacija u oblasti edukacije poslovnog odlučivanja i inteligentnih sistema. Ispunjenjem ovog cilja može se zaključiti dokle se stiglo u ovim oblastima i da li je moguće napraviti neki prodor u ovom polju.
- Razvoj novog simulatora za edukaciju inteligentnih sistema koji bi nadomestio sve nedostatke prikazanih rešenja i uveo određene nove koncepte.
- Razvoj skupa laboratorijskih vežbi koje bi se koristile za objašnjavanje određenih koncepata, a omogućili bi i da razvijeni simulator bude maksimalno iskorišćen.
- Evaluacija novog simulatora u cilju opravdavanja njegove upotrebne vrednosti u nastavi.

1.3. Hipotetički okvir istraživanja

Osnovna hipoteza je da korišćenje određenog vizuelnog softverskog simulatora može doprineti olakšanom učenju i boljem razumevanju grupi koja ga koristi. S obzirom na to da bi sa metodičke strane bio usmeren ka edukaciji i zbog toga u potpunosti prilagođen korišćenju u nastavi, grupa koja ga koristi bi trebala da postigne bolji rezultat nego da radi bez korišćenja simulatora, što se može izmeriti pomoću nekoliko ključnih faktora: *prosečna ocena, broj studenata koji je izabrao predmet i procenat studenata koji su položili ispit.*

Posebne hipoteze su:

- sve veći uticaj vizuelnih softverskih simulacija u edukaciji;
- važnost informacionih tehnologija za primene u okviru oblasti poslovnog odlučivanja;
- povećanje korisnosti postojećih softverskih simulacija u oblasti poslovnog odlučivanja razvojem novog simulatora pomoću koga bi se nadomestili nedostaci postojećih rešenja.

1.4. Metode i tok istraživačkog postupka

U izradi ovog rada korišćeno je nekoliko metoda naučnog saznanja. Upotrebljeni su:

- Metod analize sadržaja strane i domaće stručne literature i internet materijala iz oblasti softverskih simulacija i inteligentnih sistema. Na taj način, napravljen je pregled ovih oblasti, a uspostavljena je i veza između savremenog poslovanja i savremenih tehnologija. Zahvaljujući tome, došlo se do potrebnih informacija i znanja kako bi se bliže definisali osnovni pojmovi korišćeni u ovom radu, što je poslužilo kao polazna osnova za dalje proučavanje i pisanje.
- Komparativni metod koji je korišćen da bi se napravio pregled postojećih iskustava i rešenja u razvoju softvera, kao i njihovo poređenje po osnovu tehničke realizacije, funkcionalnosti i ispunjenosti funkcionalnih zahteva. Prikazano je i međusobno poređenje karakteristika izabranih simulatora.
- Metod specijalizacije, gde se pošlo od poslovnog odlučivanja preko primene informacionih tehnologija u okviru njega, do primene softverskih rešenja za edukaciju u tim oblastima.
- Metod pojedinačnog posmatranja, tj. analize slučajeva postojećih primera iz prakse. Za svaki simulator su analizirane tehničke karakteristike i funkcionalnosti, ali i faktori koji sa metodičke strane određuju kvalitetan softverski sistem.

- Statistički metodi, kao način evaluiranja upotrebljivosti novog softverskog sistema. Za prikupljanje podataka biće upotrebljeni ispitivanje i anketa, a grupa koja se ispituje će biti podeljena na kontrolnu i eksperimentalnu.

1.5. Struktura rada

Rad je podeljen na nekoliko poglavlja:

- PREGLED OBLASTI ISTRAŽIVANJA definiše mesto softverskih simulacija u savremenom obrazovanju, počevši od istorijskog pregleda do upotrebe u simulaciji složenih algoritama. Razmatra se i upotreba inteligentnih sistema u oblasti poslovnog odlučivanja, a poseban deo poglavlja je posvećen softverskim simulacijama u poslovnom odlučivanju, njihovom pregledu, kao i prednostima i manama navedenih rešenja.
- TEORIJSKE OSNOVE ISTRAŽIVANJA definišu osnovu na kojoj je započet razvoj novog simulatora. Obrazložena su uporedna iskustva i rešenja u korišćenju simulatora mašinskog učenja i pretraživanja, na osnovu čega je izvedena analiza i izbor algoritama koji će biti obuhvaćeni u simulatoru. Za svaki od izabranih algoritama definisane su teorijske osnove uz detaljan način rada i odgovarajući pregled literature.
- REALIZACIJA NOVOG SOFTVERSKOG SIMULATORA detaljnije definiše proces izrade samog simulatora. Nakon osnovnog opisa funkcionalnosti sistema, detaljno se obrazlaže način izbora tehnologija koje su korišćene za razvoj simulatora. Potom se prelazi na samu implementaciju, počevši od dijagrama klasa preko listinga sa programskim kodom koji detaljnije obrazlažu rešenja određenih problema koji su se javili tokom razvoja.
- LABORATORIJSKE VEŽBE ZASNOVANE NA SOFTVERSKOM SISTEMU definišu setove praktičnih zadataka koji se mogu koristiti u radu sa simulatorom. Ove vežbe su napravljene kako bi omogućile što bolje iskorišćenje softvera, kao i nastavnicima olakšale rad na časovima.

- EVALUACIJA REALIZOVANOG SISTEMA sumira upotrebnu vrednost novog softverskog simulatora, a sprovedena je kvalitativna i kvantitativna analiza. Faktori koji su služili kao osnova za analizu su prosečna ocena, broj studenata koji je izabrao predmet i procenat studenata koji su položili ispit.

1.6. Očekivani doprinosi

Očekivani stručni doprinosi su:

- Objašnjavanje važnosti upotrebe vizuelnih softverskih simulacija u edukaciji.
- Pregled iskustava na svetskim univerzitetima u pogledu korišćenja simulacija poslovanja, kao i pregled prednosti i mana svakog od prikazanih rešenja.
- Pregled tehnologija i biblioteka koje se mogu pogodno koristiti za razvoj softverskih simulatora.
- Predlog rešenja kojim bi se u navedenim simulacijama poslovanja korišćenjem algoritama inteligentnih sistema prevazišli problemi vezani za proces donošenja odluka - ovi problemi odnose se na vezu između velikog broja finansijskih izveštaja i pokazatelja sa jedne strane i donošenja odluka sa druge.
- Razvoj novog softverskog simulatora inteligentnih sistema koji bi koristio algoritme mašinskog učenja i algoritme pretraživanja.
- Definisanje detaljnog skupa laboratorijskih vežbi koje bi poslužile da se simulator koji je razvijen maksimalno iskoristi.

Očekivani naučni doprinosi su:

- Istraživanje se može opisati kao potreba za definisanjem novog sistema koji bi mogao da se koristi u postojećim simulacijama poslovanja kao segment koji im strukturno nedostaje i koji bi olakšao njihovo korišćenje.

- Pregled postojećih rešenja u oblasti softverskih simulacija poslovanja, algoritama mašinskog učenja i algoritama pretraživanja.
- Smernice za dalji razvoj oblasti softverskih simulacija u savremenom obrazovanju.

2. PREGLED OBLASTI ISTRAŽIVANJA

Osnovu ovog poglavlja čini pregled dosadašnjih dostignuća u oblasti poslovnih simulacija i njihova primena u savremenom obrazovanju, uz isticanje važnosti, kao i značaja njihovog korišćenja. Takođe, definiše se i veza koja postoji između savremenih poslovnih sistema za pomoć poslovnom odlučivanju i upotrebe inteligentnih sistema u okviru njih. Za razumevanje rada ovakvih sistema izuzetno su pogodne softverske simulacije - napravljen je pregled najpoznatijih rešenja za navedenu oblast uz opis i navođenje prednosti i mana za svako od njih. Kao rezultat potrebe da se primećeni problemi reše, postavljena je osnova razvoja novog simulatora koji je tema ovog rada.

2.1. Softverske simulacije u savremenoj edukaciji

Softverske simulacije zauzimaju sve važnije mesto u savremenom obrazovanju. U poslednje vreme im se posvećuje sve više pažnje, a neke oblasti se više i ne mogu zamisliti bez upotrebe ovakvog softvera. Iako postoje zapisi o borbenim igrama još u Kini, 3000. godine pre nove ere, prva moderna poslovna simulacija je predstavljena tek 1955. godine, u vreme kada informacione tehnologije doživljavaju ekspanziju [1].

2.1.1. Istorijat softverskih simulacija

Od sredine 1950.-ih godina, upotreba poslovnih simulacija je u stalnom porastu. Danas je ovaj metod izvođenja nastave dostigao visok stepen upotrebe na mnogobrojnim fakultetima. Među prvima se pojavila simulaciona vežba *Monopologs* koja se fokusirala na sistem logističke podrške vazduhoplovstva SAD-a, a razvila ju je *Rand* korporacija. Zadatak učesnika bio je da rukovode lancem snabdevanja i po strukturi je bio sličan onome što nude savremena rešenja [2]. U toku 1956. godine predstavljena je i simulacija *Top Management Decision Simulation* koja je razvijena od strane američke asocijacije za menadžment

(*American Management Association*) [3]. Potom je usledilo i predstavljanje simulatora *Business Management Game* [4].

Brz porast broja poslovnih simulacija desio se između 1958. i 1961. godine. Procenjuje se da je do 1961. godine razvijeno preko 100 rešenja i da ih je koristilo više od 30.000 rukovodilaca iz raznih kompanija. Do 1969. godine broj simulacija je porastao na oko 200. Neprestani porast je, još u to vreme, ukazivao na sve veću popularnost ove oblasti. [5]

Nakon 1980-ih godina, povećavanje kapaciteta računarskih tehnologija je dovelo do razvoja raznovrsnih problemski orijentisanih vežbi. Neke od tih vežbi su vodile studenta kroz utvrđen niz problema, koji su imali i odgovarajuće grafičke prikaze u vidu raznih dijagrama i izveštaja, zasnovanih na postojanju veza između dve ili tri promenljive. Ali, ovakve vežbe, sa strogo definisanim tokom, *ne predstavljaju* simulaciju za učesnika. Drugim rečima, kao u realnom svetu, simulacija predstavlja problem, sa *definisanim raznim parametrima i više mogućih načina delovanja*.

Brz razvoj oblasti simulacija koje se koriste u edukaciji doveo je do sve veće potražnje za složenim rešenjima. Zahvaljujući tome, danas postoji veliki broj modela za razne oblasti. Sve veći broj obrazovnih ustanova se opredeljuje da svojim studentima na ovaj način obezbedi praktično razumevanje oblasti kojima se bave. Upotreba simulacija u edukaciji ubrzava transfer iz teorije ka praksi, uz smanjivanje troškova i vremena rada [6, 7].

2.1.2. Korisnost softverskih simulacija

Simulacija se definiše kao interaktivna apstrakcija realnog života, ili kao bilo koji pokušaj da se oponaša neko okruženje ili sistem [8]. Praktično, simulacije predstavljaju vežbe u okviru kojih se moraju primeniti određena znanja, veštine i strategije u cilju ispunjavanja određenog zadatka. One predstavljaju „igre” otvorenog kraja u kojima se kroz određene situacije prolazi korišćenjem mnogobrojnih promenljivih. Potrebno je da svaki učesnik, preuzimanjem određene uloge, prouči određena stanja, pretnje ili probleme, da na osnovu toga donese određene odluke, kao i da vidi efekte svojih odluka [9, 10]. Simulacija se može odvijati u raznim pravcima, u zavisnosti od odluka koje su preduzimane.

Simulacije su posebno korisne u objašnjavanju složenih poslovnih situacija koje se mogu javiti u praksi jer predstavljaju *aktivan* edukativni alat. One predstavljaju kontrolisano okruženje u kome ne postoji rizik od pravljenja štete. Zahvaljujući tome, studenti mogu razumeti veze između svojih poteza i njihovih efekata na funkcionalne oblasti u okviru preduzeća.

Važne karakteristike vizuelnih simulacija su [11]:

- primenjivanje adekvatnog modela situacija iz realnog sveta sa kojima se učesnik suočava;
- definisana uloga za svakog učesnika, uz utvrđene odgovornosti i ograničenja;
- okruženje bogato podacima koje omogućava studentima da izvedu opseg strategija, od izuzetno širokog do vrlo prefinjenog definisanja poslovnih odluka;
- izveštaj o promenama koje su posledice akcija koje je učesnik preduzimao.

Svakako, uz odgovarajući način korišćenja, simulacije predstavljaju izvanredan metod elektronskog učenja koji čini osnovu modernog obrazovanja. Na ovaj način se može podstaći aktivniji rad studenata, pri čemu će im biti omogućeno da uče iz ličnog iskustva. Praktično, pruža im se mogućnost da se nađu u realnim situacijama, koje ih mogu očekivati i na radnom mestu. Tokom rada, nastavnik ima tačan uvid u sve aktivnosti što mu omogućava da se u svakom trenutku uključi i usmeri učesnike ukoliko naiđu na problem.

Kako bi sama simulacija bila na najbolji način prilagođena učesnicima, potrebno je voditi računa o njenoj kompleksnosti. Potrebno je uvideti vezu između složenosti igre i percepcije stvarnosti. Ukoliko je simulacija suviše jednostavna, biće shvaćena kao nerealna, pogotovo od strane učesnika koji su napredniji po nivou znanja. Sa druge strane se nalaze izuzetno složene simulacije, za koje se može reći da su izuzetno realistične. Ali, u slučajevima da je složenost na većem nivou od pripremljenosti njenih učesnika, može se raširiti stav da je nerealna zato što je izvan njihovih mogućnosti. [12] Zbog toga je prilikom projektovanja ovakve vrste softvera potrebno voditi računa o stepenu njegove složenosti i o usklađenosti sa ciljnom grupom.

Potpuno realistična simulacija ozbiljnog preduzeća bi imala ogroman broj promenljivih i stoga bi sigurno bila nepraktična. Zbog toga njihov broj treba ograničiti na neku razumnu meru. Sa druge strane, simulaciji je potrebno dati više dinamike – zbog toga se mogu postepeno uvoditi, na primer, nova tržišta ili razvoj novih proizvoda.

2.1.3. Značaj vizuelnih softverskih simulacija

Postoji drevna kineska poslovice koja glasi: „*Reci mi - zaboraviću, pokaži mi – zapamtiću, uključi me – razumeću*”. Ova tvrdnja se pokazuje posebno istinitom u oblasti kao što je poslovno odlučivanje. Korisnici mogu razumeti kako određena tehnika radi ukoliko im se da odgovarajuća literatura, ili na tabli ispiše postupak rešenja nekog primera. Ali, ukoliko im se omogući da to sami isprobaju i da se uključe, onda mogu razumeti mnogo lakše, ali i bolje.

Vizuelne softverske simulacije u poslovnom odlučivanju su značajan metod praćenja i razumevanja poslovnih procesa u modernom preduzeću. Zahvaljujući njima, podstiče se razvijanje preduzetničkih stavova i učenje tehnika savremenog menadžmenta. Za studente je stvorena mogućnost da postanu, na primer, poslovni menadžeri koji moraju da donose odluke u strogim tržišnim uslovima. Preuzimanjem takve uloge, učesnik se nalazi u situaciji u kojoj je neophodno preuzeti finansijski rizik sopstvenih odluka - zbog ovog razloga je korišćenje simulacija značajno. [13]

Istraživanje metoda edukacije je pokazalo da učenje bazirano na praktičnoj primeni znanja i treniranje u okruženju koje je što približnije realnom daje najbolje rezultate. U mnogim poljima aktivnosti, kao što je rukovođenje preduzećem, korišćenje simulacija je jedan od najpopularnijih načina učenja. Na ovaj način za učesnike se obezbeđuje brz transfer znanja iz simuliranog okruženja na realne situacije koje ih očekuju na poslu.

U sledećoj listi su prikazane koristi korišćenja simulacija kao nastavnog alata [13]:

- zanimljiv način učenja jer studenti mogu prikupljati i ispitivati znanje tokom rada sa simulacijom;
- ubrzavanje procesa učenja aktivnim angažovanjem učesnika;
- mogućnost praćenja napretka u poboljšavanju sopstvenih znanja i veština;

- povezivanje znanja iz različitih oblasti;
- jaka motivacija učesnika ka aktivnom učenju umesto ka pasivnom praćenju nastavnika;
- Oblikovanje veština korišćenja stečenog znanja u praksi;
- interaktivni karakter simulacije koji omogućava učesnicima da brzo prikupljaju informacije o rezultatima svojih odluka.

Takođe, isti autori prikazuju i neke od negativnih strana korišćenja simulacija:

- scenario simulacije je model realnog sveta, što znači da je ipak napravljen uz određena pojednostavljenja;
- učenje je ograničeno samo na određenu oblast znanja i veština koje su bile uzete u obzir prilikom pravljenja simulacije;
- uvek postoji barijera od strane određene grupe ljudi koji smatraju da je simulacioni rad igranje;
- zbog nepostojanja posledica izvan simulatora za preduzete odluke, ponašanje učesnika se može razlikovati od onog u stvarnom životu.

U današnje vreme, simulacije zauzimaju sve važnije mesto u nastavi [14]. Preporučuje se korišćenje odgovarajućeg softvera na laboratorijskim vežbama u oblasti računarskog inženjerstva kao izuzetno važnog načina da se studentima omogući da prate, istražuju i rukuju karakteristikama i ponašanjima uređaja, sistema i procesa. [15, 16] Preporučuje se korišćenje aplikacija i simulacija u cilju modelovanja i proučavanja realnih sistema koje nisu praktični za fizičku implementaciju. [17] U oblasti računarskog inženjerstva do sada je razvijen izuzetno veliki broj simulatora. Na primer, simulacije se koriste za učenje računarske arhitekture i organizacije [18], ili kao dodatni alat za kurseve u oblasti elektronike. [19] Svi ovi navodi doprineli su u tome da se započne sa realizacijom simulatora inteligentnih sistema koji je tema ovog rada.

U skladu sa prirodom informacionih tehnologija i očekivanja studenata koji pohađaju takve kurseve, softver ima najvažniju ulogu. Softver se može koristiti za [20]:

- motivisanje studenata;
- usmeravanje njihovog razmišljanja i ambicija;
- objašnjavanje značaja određenih materijala;
- demonstraciju primene teoretskih ideja.

Na ovaj način, studenti treba da se motivišu praktičnom stranom inženjerstva. U osnovi, pažljivo isplanirani praktični zadaci u laboratoriji bi trebali da pomognu u razvijanju poverenja u sopstvene sposobnosti.

Još jedna važna korist koju ovakav softver može imati jeste podrška učenju na daljinu. Tehnike i algoritmi koji se koriste u informacionim tehnologijama nekad mogu biti komplikovani za razumevanje, pogotovo ako se proučavaju iz pisane literature. Mnogo lakši način za razumevanje može obezbediti interaktivna demonstracija, čiji je tok čak moguće i kontrolisati. Na taj način, moguće je postupno pratiti šta se dešava, a po potrebi se i vratiti jedan ili više koraka unazad kako bi se ponovo sagledao određeni segment. Na taj način, studenti i van fakulteta mogu samostalno pratiti i lakše razumeti određenu oblast.

2.2. Inteligentni sistemi u poslovnom odlučivanju

2.2.1. Proces donošenja odluka

Donošenje odluka čini deo svakodnevnog života. Neke odluke su izuzetno visokog stepena važnosti (npr. važne poslovne odluke), dok su druge krajnje jednostavne – kao odluka o tome šta jesti za večeru. Sve odluke zahtevaju određeni proces razmišljanja uz korišćenje određenih informacija – ali ne u istom stepenu. Racionalno razmišljanje govori da bi važnijim odlukama trebalo posvetiti više vremena, ali to nije uvek slučaj jer donošenje odluka nije uvek racionalan proces. To je proces kojim upravlja naš sistem vrednosti i percepcija nesigurne budućnosti. Nekada može doći do zastoja u odlučivanju kada prevlada strah od neuspeha i sećanje na kajanje posle nekih prethodnih odluka.

Psihološka i mentalna zrelost osobe su povezane sa sposobnošću donošenja razumnih odluka, preuzimanja rizika, tolerisanja grešaka i preuzimanja odgovornosti za posledice donetih odluka. Ne postoji način da se izbegne osećaj kajanja posle donošenja određene odluke koji smo svi iskusili u nekom trenutku. Iskusni donosioci odluka izvlače korist od pogrešnih odluka – oni uče iz njih. Na taj način postajemo svesni svojih vrednosti, naše percepcije stvarnosti i na taj način učimo da se menjamo. Osoba koja je efektivan donosilac odluka uči iz iskustva i menja se kada se promeni opažanje eksternih uslova. [21]

Proces donošenja odluka je generički proces koji se može primeniti na sve forme organizovanih aktivnosti, a koristi se već vekovima. Većina ljudskih, kao i svaka inženjerska aktivnost, na direktan ili indirektan način, imaju neke svoje ekonomske ciljeve. Ostvarivanje tih aktivnosti je direktno povezano sa donošenjem niza određenih odluka. Zbog toga se proces odlučivanja proučava na mnogim univerzitetima, ali i u velikim poslovnim sistemima. [22] Teorija odlučivanja je postala važna naučna disciplina , posebno važna za zaposlene u ekonomiji, inženjerskim naukama, politici, medicini, pravu, itd. Ona zahteva pravljenje izbora. Vršenje izbora zahteva obavljanje određenog procesa. Proces odlučivanja se može odraziti na sve modele ljudske svesnosti koji obuhvataju ljudska opažanja, osećaje i logiku. Opažanja omogućuju prevođenje podsticaja okoline u apstraktne modele, osećaji nas vode ka izboru pravih vrednosti i odgovarajućih kriterijuma, a logika upućuje na racionalni proces izbora akcije (alternative) u cilju realizovanja određenih ciljeva. Teorija odlučivanja obezbeđuje okvir za delovanje, koji dozvoljava doprinos sva tri pomenuta aspekta ljudskog saznanja. [22]

Informacije obezbeđene proučavanjem matematičkih modela predstavljaju važan faktor u procesu odlučivanja. Treba naglasiti da model ni na koji način ne treba da kaže šta treba da se odluči. Modeli teorije odlučivanja nisu namenjeni da zamene intuitivno rasuđivanje – oni ga dopunjuju, ali ga ne mogu isključiti. Kritičko rasuđivanje je potrebno za definisanje ciljeva, identifikovanje i stvaranje alternativa, u izboru važnih kriterijuma, kao i u radu sa svim onim faktorima koji se ne smeju prepustiti samo kvantitativnoj proceni. Izbor karijere, kupovina kuće, odlučivanje o medicinskom tretmanu ili poslovne odluke su složene situacije. Pri suočavanju sa takvim situacijama, u određenim slučajevima se mogu koristiti modeli teorije odlučivanja. [21]

2.2.2. Uloga informacionih tehnologija u savremenom poslovnom odlučivanju

Uspeh upravljanja organizacijom zavisi od veštine obavljanja funkcija upravljanja kao što su planiranje, organizovanje, rukovođenje i kontrola. Da bi se izvršile ove funkcije, menadžeri učestvuju u stalnom procesu donošenja odluka. [23]

Veoma je teško donositi dobre odluke bez dobrih informacija. Informacije su potrebne za svaku fazu i aktivnost u procesu donošenja odluka. Ukoliko bi se informacije ručno obrađivale, proces bi suviše dugo trajao, a informacije su često potrebne u veoma kratkim vremenskim rokovima. Zbog toga, sistemi za pomoć u poslovnim odlučivanju sve više dobijaju na važnosti.

Budući da se savremeno poslovanje u velikoj meri oslanja na korišćenje savremenih tehnologija, ogromne količine podataka se prikupljaju iz poslovnih transakcija. Kako bi se ovi podaci prilagodili i postali upotrebljivi za donošenje odluka, koriste se tehnike *iskopavanja podataka* (*data mining*). Iskopavanje podataka se zasniva na računarski orijentisanom pretraživanju i analizi podataka u cilju pronalaženja korisnih obrazaca među podacima. Ovi obrasci predstavljaju novo znanje pomoću koga se mogu unaprediti marketing aktivnosti, prodaja, odnosi sa kupcima, kao i donošenje odluka. Na ovaj način se može doći do strateški važnih podataka o kupcima i njihovim interesovanjima. Za potrebe pretraživanja podataka koriste se algoritmi mašinskog učenja.

Kompanije idu i korak dalje u prikupljanju informacija o kupcima pa se prilikom kupovine generišu informacije o proizvodu koji je potrošač naručio, katalogu iz kojeg je proizvod izabran, broju platne kartice, veličini porudžbine i vremenskom intervalu između prethodne i trenutne narudžbine. Dalje se uzimaju u obzir način plaćanja i dostave proizvoda, eventualne reklamacije, tehnička podrška i pitanja u vezi sa korišćenjem proizvoda. Transakcije obavljene putem interneta mogu da pruže još više informacija, s obzirom da potrošač prilikom onlajn kupovine ostavlja niz podataka o sebi i svojim navikama, a podaci u vezi sa transakcijama potrošača lako mogu biti povezani i sa podacima o ponašanju prilikom pretrage veba. Niz podataka o istoriji kupovine putem interneta, finansijskoj istoriji i drugi lični podaci o potrošaču samo su nekoliko klikova mišem daleko. Kao što je navedeno, prvi

korak procesa iskopavanja podataka je prikupljanje ovakvih podataka o ponašanju potrošača, bilo da su izvor internet transakcije, kupovina u prodavnici ili se koriste neki drugi izvori podataka o korisniku. [24]

Sve jeftinija upotreba informacionih tehnologija je u velikoj meri uticala na proces korporativnog donošenja odluka. Pristup modernim bazama podataka i naprednim softverskim rešenjima omogućio je znatno veće mogućnosti u oblasti odlučivanja. Širenjem sistema za elektronske i mobilne komunikacije, donosioci odluka još brže mogu dobiti informacije, a takođe, mogu i veoma brzo delovati i realizovati svoje odluke.

2.2.3. Sistemi za podršku odlučivanju

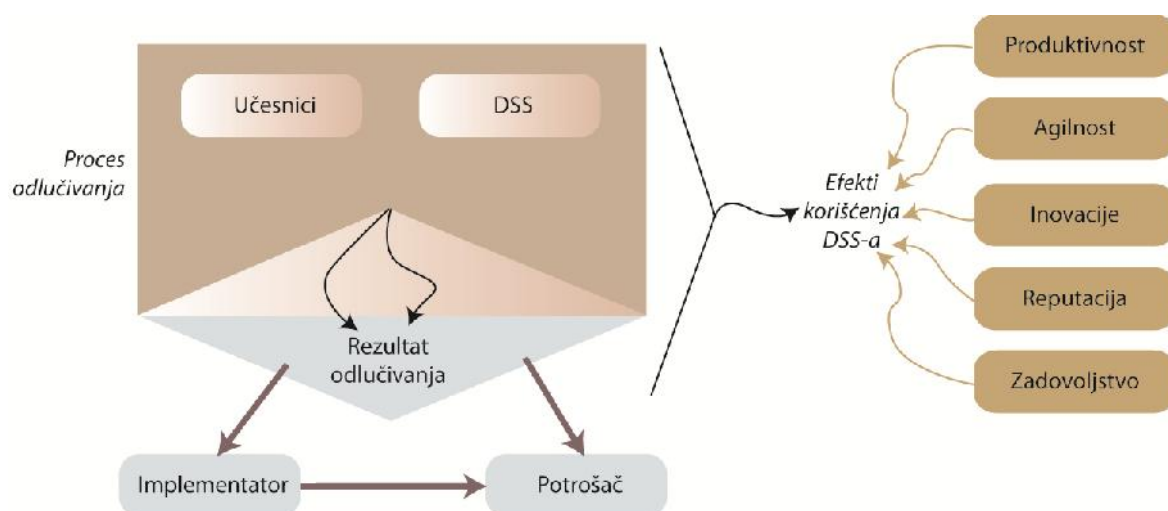
S obzirom da je kvalitet poslovnih odluka koje treba doneti od izuzetno velike važnosti, proces odlučivanja je u nauci kroz istoriju zauzimao važno mesto. U okviru statistike, ekonomije i operacionih istraživanja razvijani su razni metodi za pravljenje racionalnih izbora. U poslednje vreme, ovi metodi su poboljšani raznovrsnim tehnikama poreklom iz informacionih tehnologija i veštačke inteligencije implementiranim u obliku računarskih programa, bilo kao samostalni alati ili kao deo integrisanih okruženja za donošenje složenih odluka. Ovakvi programi se jednim imenom nazivaju sistemi za podršku odlučivanju (engl. Decision Support System - DSS). Ovaj koncept je izuzetno širok, a grubo se mogu definisati kao interaktivni računarski sistemi koji pomažu korisnicima u rasuđivanju i prilikom pravljenja izbora.

Sistemi za podršku odlučivanju stiču sve veću popularnost u raznim oblastima, kao što su poslovno okruženje, medicina, inženjerske i vojne primene. Posebno su važni u situacijama kada količina dostupnih informacija onemogućava korišćenje intuicije donosioca odluka, kao i kada su brzina i preciznost odlučivanja od velike važnosti. Ovakvi sistemi mogu integrisati razne izvore informacija, obezbeđujući inteligentni pristup znanju, i pritom pomoći prilikom strukturiranja odluka. Takođe, mogu se i upotrebiti tehnike veštačke inteligencije kako bi se obradili heuristički problemi koji su nepogodni za klasične tehnike. Odgovarajuća primena alata poslovnog odlučivanja povećava produktivnost, efikasnost i efektivnost i daje mnoge komparativne prednosti u odnosu na konkurenciju. Na taj način, omogućeno je pravljenje

optimalnih izbora za tehnološke procese i njihove parametre, planiranje poslovnih operacija, logistike ili investicija. [25]

Proces donošenja odluke se može predstaviti crnom kutijom, a može uključivati akcije DSS-a, kao i drugih učesnika. Finansijer, učesnici, implementator i potrošač mogu biti različite osobe ili jedna osoba može imati više uloga (grafikon 1) [26]:

Grafikon 1: Uloga sistema za podršku odlučivanju



Prilagođeno prema: Holsapple W. C., *DSS Architecture and Types, Handbook on Decision Support Systems 1*, Springer-Verlag, Heidelberg, 2008, str. 164.

Kada je DSS (ili više njih) uključen u proces donošenja odluke, on utiče na taj proces i na njegov izlaz kroz najmanje jedan od pet faktora – produktivnost, agilnost, inovacije, reputaciju i zadovoljstvo. Jasno se vidi da je odnos svih elemenata prema korisniku jednostavan i jasan.

2.3. Postojeća rešenja u edukaciji poslovnog odlučivanja

Do sada je razvijeno mnoštvo ovakvih rešenja, a za potrebe ovog rada, odabrana su samo ona najpotpunija koja su i najviše zastupljena u svetu. Ovi simulatori se koriste na velikom broju univerziteta, gde dokazuju svoju upotrebnu vrednost. U narednim odeljcima svaki od njih je detaljnije predstavljen.

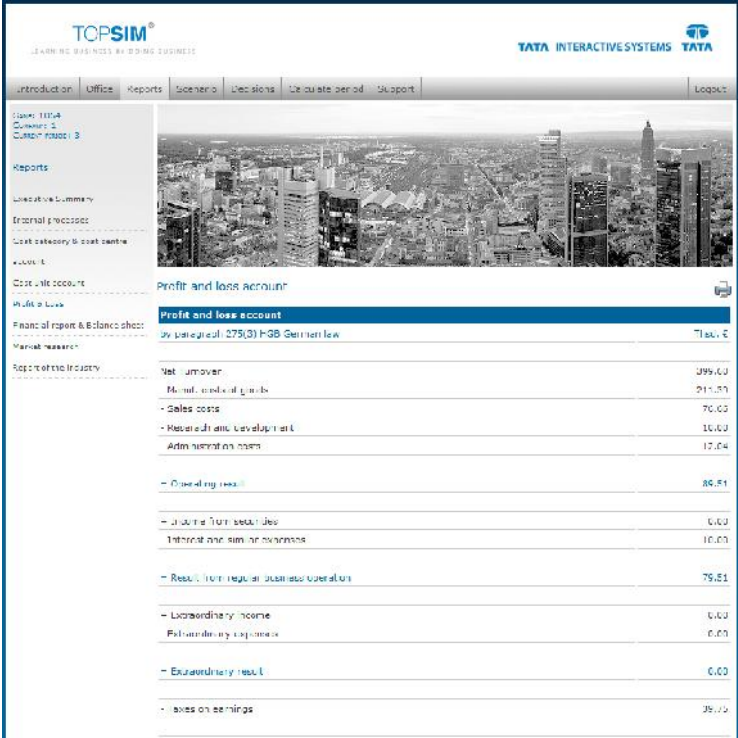
2.3.1. TopSim Easy Management

Simulacija *TopSim Easy Management* (<http://www.topsim.com>) podrazumeva preuzimanje upravljanja porodičnom firmom koja se bavi proizvodnjom opreme za rekreaciju. Potrebno je razviti poslovnu ideju, napraviti proizvod i sprovesti odgovarajuće marketing aktivnosti. U okviru simulacije, moguće je uticati na politiku cena robe, marketing, broj zaposlenih, proizvodnju, istraživanje i razvoj... Potrebno je napraviti balans između svih ovih parametara, kako bi preduzeće moglo da ostvari dobit. Zbog toga, postoje bilans stanja i uspeha, veliki broj izveštaja o prodaji, zaliham materijala, opterećenju pogona... Kako bi bilo moguće doneti pravu odluku, potrebno je protumačiti ove izveštaje.

Ovakvo rešenje omogućava učesnicima razumevanje osnovnih ekonomskih i preduzetničkih znanja i motiviše ih da donose korporativne odluke. Takođe, omogućeno je transparentno pregledanje svih efekata koje je određena odluka proizvela.

Grafikon 2 prikazuje izgled ekrana simulacije TopSim Easy Management u okviru koga se vidi samo jedan od izveštaja.

Grafikon 2: Početna strana simulacije TopSim Easy Management



The screenshot displays the TopSim Easy Management web application. The interface includes a navigation menu on the left with options like 'Introduction', 'Office', 'Reports', 'Scenarios', 'Decisions', 'Calculate period', and 'Support'. The main content area shows a 'Profit and loss account' report for the period 27/03/2014 to 31/03/2014. The report is presented in a table format with various financial metrics and their corresponding values.

Profit and loss account	
Net turnover	1099.00
- Maint. material goods	211.50
- Sales costs	70.00
- Research and development	10.00
- Administration costs	17.04
= Operating result	89.46
- Financial result (excluding interest)	0.00
= Result from regular business operations	89.46
- Extraordinary income	0.00
- Extraordinary expenses	0.00
= Extraordinary result	0.00
- Taxes on earnings	35.00

2.3.2. Harvard Working Capital Simulation

Na Univerzitetu Harvard (<http://hbsp.harvard.edu/>) razvijeno je više onlajn simulacija, a jedna od njih je i *Working Capital Simulation* (grafikon 3). Učesnici preuzimaju ulogu rukovodioca kompanije *Sunflower Nutraceuticals*, a cilj je da pažljivo investiraju sredstva, kao i da iskoriste prilike pomoću kojih mogu poboljšati tok novca. Svaka od tih prilika predstavlja zaseban zadatak koji je potrebno pažljivo analizirati, kako bi se videlo kakve će efekte ostvariti na poslovanje. Potrebno je pažljivo izbalansirati želju za rastom, uz očuvanje likvidnosti. Ova simulacija je pogodna da se koristi na uvodnim kursevima iz finansija, računovodstva ili preduzetništva. Problem može predstavljati trajanje – cela simulacija se može završiti veoma brzo, tako da je nije moguće koristiti tokom većeg broja časova.

Grafikon 3: Prikaz toka novca u simulaciji *Working Capital Simulation*



2.3.3. Cesim GlobalChallenge

Simulacija *GlobalChallenge* kompanije *Cesim* (<http://globalchallenge.cesim.com/>) osmišljena je da poboljša razumevanje poslovanja u dinamičnom, takmičarskom okruženju, kao i da omogući da studenti razumeju koncepte strategijskog menadžmenta i internacionalnog poslovanja (grafikon 4). Pred studente se stavlja zadatak vođenja globalne telekomunikacione kompanije kroz tehnološku i marketinšku evoluciju. Potrebno je da razviju poslovne strategije za svoju kompaniju koja posluje na teritoriji Evrope, SAD i Azije. Moguće je uticati na razne promenljive u oblasti ekonomije, finansija, ljudskih resursa, računovodstva, proizvodnje, marketinga, logistike, istraživanja i inovacija. Cilj je poboljšati stepen razumevanja složenosti globalnog poslovanja, kao i shvatanja uticaja koji ima donošenje poslovnih odluka.

Ova simulacija je dostupna na više jezika, među kojima je i srpski, što svakako predstavlja veliki plus.

Grafikon 4: Početna strana simulacije Cesim GlobalChallenge

The screenshot displays the Cesim GlobalChallenge web interface. At the top, there is a navigation bar with links: 'početna strana', 'odlike', 'rezultati', 'raspored', 'timovi', 'uputstva', and 'forumi'. The main content area is titled 'Početna strana' and features a 'Dream Team 3' profile. The profile includes a user icon, name 'Gary Player', email 'player1@cesim.com', organization 'Cesim', time zone 'CET', and country 'Finland'. Below this, there is a section for 'Vaš instruktor' (Your instructor) listing 'Test Instruktor' and 'Ylli Tuovinen'. A 'Članovi Vašeg tima' (Members of your team) section shows 'Gary Player' as an online member. The 'Evidencija odluka tima' (Team decision log) table at the bottom has columns for 'SACUVANO NA', 'SACUVANO A', 'OD UKUP. IF. KONFER. A', 'OD UKUP. TIM. IF. KONFER. A', and 'DNEVANJE DANA'. The right sidebar contains 'Podaci o tržištu za prethodnu rundu' (Market data for the previous round) with statistics like 'Prosečna u prihodima od prodaje' (16,1 %), 'Prosečna u prosečnim rezultatima' (48,2 %), and 'Najveća dobit' (1 284 408). It also lists 'Najbliži rok' (Next deadline) as 'ROK ZA -' and 'Trenutno vreme' (Current time) as 'Vaše vreme 2012-10-18 11:31'. A 'Forum' section shows '5 novih poruka na forumu Vašeg tima' and a 'Kviz' (Quiz) section with 'Nema kvizova'.

2.3.4. Forio Simulate

Forio Simulate (<https://forio.com/>) predstavlja veoma zanimljiv pristup oblasti vizuelnih simulacija. Pred korisnika je stavljen na raspolaganje kompletan mehanizam razvoja sopstvene simulacije. Počinje se od definisanja modela, a moguće je kao osnovu koristiti i Microsoft Excel. nakon toga se definišu jednačine koje opisuju poslovne procese. Ukoliko se vrši uvoz iz Excel-a, veliku olakšicu će predstavljati činjenica da će sve formule biti sačuvane i ugrađene u model simulacije – ukoliko se javi potreba, kroz odgovarajući dijalog ih je moguće izmeniti ili ukloniti. Takođe, potrebno je definisati i promenljive koje će biti moguće menjati kako bi se u simulaciji mogle donositi poslovne odluke. Nakon definisanja matematičke osnove sistema, moguće je pregledati grafički model simulacije (grafikon 5), a potom napraviti i korisnički interfejs. Sistem rada je sličan onom u grafičkim razvojnim okruženjima – potrebno je iz liste komponenti izabrati odgovarajuću, a potom je povezati sa modelom simulacije. Bitno je naglasiti da je podržan rad više učesnika u isto vreme.

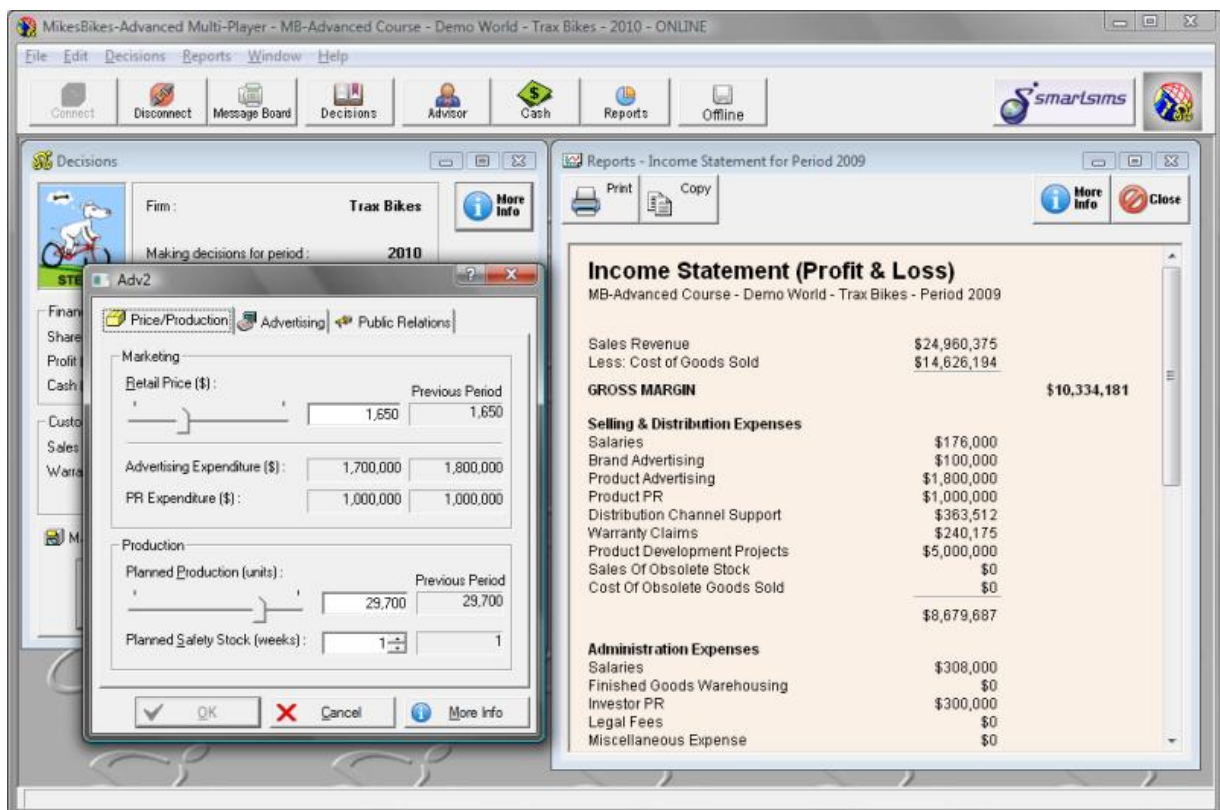
Grafikon 5: Izgled modela simulacije u okviru Forio Simulate



2.3.5. SmartSims MikesBikes-Advanced

Simulacija *SmartSims MikesBikes-Advanced* (<http://www.smartsims.com>) omogućava razumevanje veze između raznih oblasti poslovanja, uz pokazivanje na koji način razvoj i uvođenje poslovne strategije utiče na ove oblasti (grafikon 6). Učesnici uzimaju uloge menadžment tima proizvođača bicikala i dobijaju mogućnost da donose sve ključne odluke koje se tiču definisanja cene, marketing aktivnosti, distribucije proizvoda, finansija, ljudskih resursa i istraživanja i razvoja. Dostupni su detaljni izveštaji o poslovanju na osnovu kojih je potrebno donositi odluke. U toku simulacije se pred učesnike stavljaju novi izazovi, kao što je razvijanje novih proizvoda i otvaranje novih tržišta. Takođe, učesnici se takmiče između sebe tako da je potrebno stalno analizirati promene na tržištu i shvatati potrebe kupaca. Interesantna mogućnost je preuzimanje ili ujedinjavanje sa konkurentskim preduzećem, što unosi potpuno novu dinamiku u ceo proces učenja.

Grafikon 6: Izgled ekrana u simulaciji *SmartSims MikesBikes-Advanced*



2.3.6. JA Titan

U okviru simulacije *JA Titan* (<http://titan.ja.org/>) studenti upravljaju proizvodnom kompanijom, a moguće je uticati na mali skup parametara u vezi sa proizvodnjom, marketingom, investicijama, ali i dobrotvornim radom. Cilj je ostvariti što veći „indeks uspešnosti“. Ovaj pokazatelj predstavlja kombinaciju ostvarenih vrednosti profita, potencijala ponude i tražnje na tržištu, produktivnosti, tržišnog udela i ostvarenog rasta. Učesnik koji na kraju simulacije ostvari najveći indeks uspešnosti ostvaruje pobeđu. Iako je ceo sistem dobro zamišljen, u određenim segmentima pokazuje nedostatak složenosti, što ga svrstava u simulacije koje se mogu koristiti na uvodnim predmetima u oblasti preduzetništva, finansija ili marketinga. Grafikon 7 prikazuje izgled simulacije *JA Titan*.

Grafikon 7: Izgled jednog od ekrana simulacije *JA Titan*



2.4. Definisanje problema i predlozi za poboljšanja korišćenjem inteligentnih sistema

Sa aspekta izučavanja oblasti poslovnog odlučivanja, kao i algoritama koji se koriste u okviru nje, prikazani simulatori, iako veoma složeni, ne predstavljaju u potpunosti pogodno rešenje jer ne poseduju mogućnost potpunog objašnjavanja procesa donošenja odluka. U svakom od njih postoji samo mogućnost dobijanja detaljnih izveštaja i bilansa, na osnovu kojih je potrebno doneti odluke o ulaganjima, prodaji, cenama proizvoda itd. Problem se nalazi između ove dve faze - korišćenja izveštaja i donošenja odluka. Ne postoji bilo kakva mogućnost da korisnici detaljnije uđu u ovaj proces i dobiju pomoć u analiziranju mnoštva raznih parametara.

Praktično, nakon proučavanja parametara, potrebno je samostalno doneti odluke, a da pritom ne postoji način da se ceo taj proces učini lakšim za razumevanje. Ovo je posebno nepogodan pristup u situacijama kada postoji mnogo vrednosti koje predstavljaju ulaz, a za donošenje odluka su važne samo neke od njih, dok se ostale u određenim situacijama mogu zanemariti. Nijedno od prikazanih rešenja ne poseduje ovako važnu mogućnost. S obzirom da se u današnje vreme menadžeri suočavaju sa ogromnim količinama informacija među kojima moraju izabrati samo one najvažnije, to može predstavljati veliki nedostatak.

Na ovom polju, kao odlično rešenje mogu se koristiti algoritmi inteligentnih sistema. Oni omogućavaju da se prevaziđu svi navedeni problemi, kao i da se omogući predstavljanje skupa metoda pomoću kojih računari trebaju da podrže rešavanje problema putem analiziranja (učenja) slučajeva koji su se već dogodili. U cilju njihovog lakšeg razumevanja, moguće ih je predstavljati i grafički, kako bi korisnici imali pregled aktivnosti koje se dešavaju.

Korišćenje algoritama inteligentnih sistema i njihovo grafičko predstavljanje su segmenti koji nedostaju predstavljanim simulatorima, što bi kod studenata poboljšalo razumevanje oblasti poslovnog odlučivanja. Takođe, u cilju lakšeg razumevanja bi bilo potrebno

obezbediti da se kroz navedene algoritme može kretati korak po korak, kao i vraćati se unazad, kako bi njihov način rada mogao da se ispituje detaljno. To bi svakako upotrebljivost moglo da digne na znatno viši nivo, a na žalost, nijedan od prikazanih sistema ne omogućuje ovakav način rada u bilo kom svom segmentu.

3. TEORIJSKE OSNOVE ISTRAŽIVANJA

U okviru ovog poglavlja je prikazana teorijska osnova rada. Dat je pregled postojećih dostignuća u oblasti razvoja simulatora mašinskog učenja i algoritama pretraživanja. Napravljen je i pregled predmeta koji se bave inteligentnim sistemima na vodećim svetskim univerzitetima, kao i tema koje se u okviru njih obrađuju. Obuhvaćeni su i predlozi vodećih strukovnih organizacija, kao što su IEEE i ACM. Na osnovu iznetih zaključaka, postavljena je osnova simulatora koji je tema ovog rada.

3.1. Faktori upotrebljivosti određenog softvera

Kako bi neki softver mogao da se smatra upotrebljivim, mora na prvom mestu imati korisnika, a ne sam sistem. Od samog početka, počevši od faze planiranja softvera, potrebno je u centar staviti korisnika i njegove potrebe. Korisnici jednostavno žele da obave određeni zadatak i zbog toga je potrebno razvoj podrediti njihovim zahtevima.

Prema preporukama organizacije ACM, postoji ukupno šest faktora koji opisuju korisnost određenog softvera. To su [27]:

- funkcionalnost;
- lakoća učenja;
- efikasnost obavljanja zadataka;
- lakoća pamćenja;
- lično zadovoljstvo;
- razumljivost korišćenja.

Funkcionalnost podrazumeva da program može zadovoljiti sve zadatke koji bi korisniku mogli da zatrebaju. *Lakoća učenja* obuhvata stepen lakoće učenja korišćenja sistema za razne kategorije korisnika. *Efikasnost obavljanja zadataka* se bavi efikasnošću programa u slučajevima veoma česte upotrebe. *Lakoća pamćenja* pokazuje koliko lako se povremeni

korisnik može priviknuti i snaći u korišćenju. *Lično zadovoljstvo* se odnosi na to koliko koliko je korisnik zadovoljan sistemom. *Razumljivost korišćenja* pokazuje koliko lako je razumeti šta sistem zapravo radi. Ovaj faktor je posebno važan u neobičnim situacijama, npr. ukoliko se desila neka greška. Samo razumevanje toga šta sistem radi u tom trenutku može pomoći korisniku. [27]

U praksi, praktično je nemoguće zadovoljiti sve ove uslove. Najčešće je potrebno doneti odluku u kom stepenu bi svaki od navedenih faktora trebalo ispuniti. Na primer, ako se radi o elektronskoj prodavnici, cilj bi bio maksimalno usmeravanje ka lakoći učenja i ličnom zadovoljstvu u cilju privlačenja što većeg broja kupaca. Sa druge strane, sistem čiji je posao kontrola vazdušnog saobraćaja bi trebao da bude usmeren ka efikasnosti obavljanja zadataka i razumljivosti korišćenja.

Ukoliko se ovi faktori pogledaju u kontekstu razvoja softverskog simulatora, bilo bi potrebno ispuniti nekoliko faktora. Funkcionalnost programa i lakoća njegovog učenja se podrazumevaju kako bi korisnik mogao što lakše da se snađe u korišćenju softvera i da ga iskoristi na što bolji način u skladu sa svojim potrebama. Lakoća pamćenja i lično zadovoljstvo su svakako faktori koji bi trebali da budu zastupljeni u što većoj meri. Možda najvažnija stavka koju bi trebalo ispuniti jeste razumljivost korišćenja. S obzirom na to da je namena prvenstveno edukativna, razumevanje toga što se u svakom trenutku dešava u pozadini svakako je od izuzetno velike važnosti.

3.2. Uporedna iskustva i rešenja u korišćenju simulatora inteligentnih sistema

Organizacija *ACM* u okviru liste oblasti koje se izučavaju na diplomskim studijama računarskih nauka sadrži i Inteligentne sisteme. U okviru ove oblasti, posebna pažnja je posvećena mašinskom učenju, a posebno se savetuje obrađivanje stabala odlučivanja, statističkog obučavanja, verovatnosnog rasuđivanja i neuronskih mreža [28, 29]. Takođe, savetuje se i obrađivanje algoritama pretraživanja. Između ostalog, predlaže se i korišćenje pretrage u dubinu i u širinu, algoritma A^* , kao i pohlepni algoritama. Jasno se vidi da se algoritmi koji su odabrani u ovom radu u potpunosti uklapaju u ove preporuke.

Na međunarodnoj konferenciji o tehnikama *Data Mining*-a održanoj 2006. godine, čiji je organizator IEEE (*Institute of Electrical and Electronics Engineers*), identifikovano je 10 najboljih algoritama u ovoj oblasti [30]:

- Stabla odlučivanja;
- K-Means;
- SVM;
- Apriori;
- EM;
- PageRank;
- AdaBoost;
- KNN;
- Naive Bayes;
- CART.

Na mnogim univerzitetima u svetu, obrađuju se algoritmi mašinskog učenja i pretraživanja. U narednoj listi navedeni su neki od najpoznatijih, uz listu predmeta na kojima se koriste:

- Na Univerzitetu Berkli, u okviru predmeta *Uvod u veštačku inteligenciju* (*Introduction to Artificial Intelligence*), detaljno se obrađuju svi algoritmi mašinskog učenja obrađeni u ovom radu, što se detaljno vidi iz silabusa ovog predmeta [31]. Još veću težinu ovakvom izboru daje i činjenica da je za ovaj predmet i njegovu organizaciju zadužen profesor Stjuart Rasel, koautor referentnog udžbenika iz ove oblasti u svetu – *Veštačka inteligencija - moderni pristup* [70] – koji predstavlja osnovni udžbenik iz oblasti veštačke inteligencije, mašinskog učenja i algoritama pretraživanja na više od hiljadu obrazovnih institucija u preko sto zemalja.
- U okviru predmeta *Mašinsko učenje* (*Machine Learning*) na Univerzitetu Stenford, između ostalog, detaljno se obrađuju klasterovanje, Naive Bayes i neuronske mreže [32]. Na predmetu *Algoritmi: Dizajn i analiza 1 i 2* (*Algorithms: Design and*

Analysis 1 & 2) obrađuju se pretraga u dubinu i širinu, pohlepni algoritmi, kao i heuristički pristup pretraživanju [33, 34].

- Na MIT-u se u okviru predmeta *Mašinsko učenje (Machine Learning)* obrađuju svi algoritmi mašinskog učenja korišćeni u ovom radu, a posebna pažnja je posvećena Bajesovim mrežama, klasterovanju i neuronskim mrežama [35]. Stabla odlučivanja su izdvojena i obrađuju se u okviru predmeta *Veštačka inteligencija (Artificial Intelligence)* [36]. Na predmetu *Dizajn i analiza algoritama (Design and Analysis of Algorithms)* obrađuju se svi najvažniji algoritmi pretraživanja, uključujući i one predstavljene u ovom radu [37].
- Na Univerzitetu Karnegi-Melon, na predmetu *Mašinsko učenje (Machine Learning)* obrađuje se nekoliko algoritama iz ove oblasti. Među njima se nalaze sva četiri algoritma mašinskog učenja obrađena u ovom radu [38].
- Na Univerzitetu Mاستriht u Holandiji u okviru predmeta *Inteligentne pretrage i igre (Intelligent Search and Games)* obrađuju se svi algoritmi pretraživanja koji su tema ovog rada [39]. U okviru predmeta *Data Mining* obrađuju se mnogobrojni algoritmi pretraživanja [40].
- Na Elektrotehničkom fakultetu u Beogradu na predmetu *Ekspertski sistemi* obrađuju se između ostalog strategije pretraživanja, predstavljanje znanja u formalnoj logici, rasplinuta logika i strategije rešavanja problema, u okviru kojih se obrađuju stabla odlučivanja [41]. Takođe, pokrivene su i strategije pretrage, a obuhvaćeni su svi algoritmi pretraživanja koji su upotrebljeni i u ovom radu.

U okviru narednih sekcija biće prikazan pregled postojećih simulatora mašinskog učenja i algoritama pretraživanja koji su tema ovog rada. Biće napravljen pregled najpoznatijih simulatora koji se već koriste na univerzitetima širom sveta. Za svaki od njih biće naveden opis funkcionalnosti, kao i najvažnije prednosti i mane.

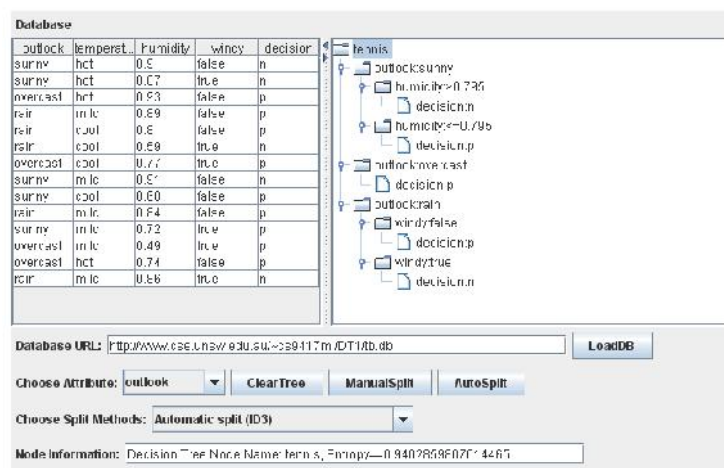
U okviru svih navedenih rešenja, postoji veoma malo sistema u okviru kojih je moguće korak po korak proći kroz određeni algoritam i dobiti rešenje. To su obično aplikacije u

okviru kojih je obrađen samo jedan algoritam, ili sistemi kojima primarni cilj nije edukacija, pa stoga nisu pogodni za korišćenje u nastavi.

3.2.1. Postojeći simulatori algoritama mašinskog učenja

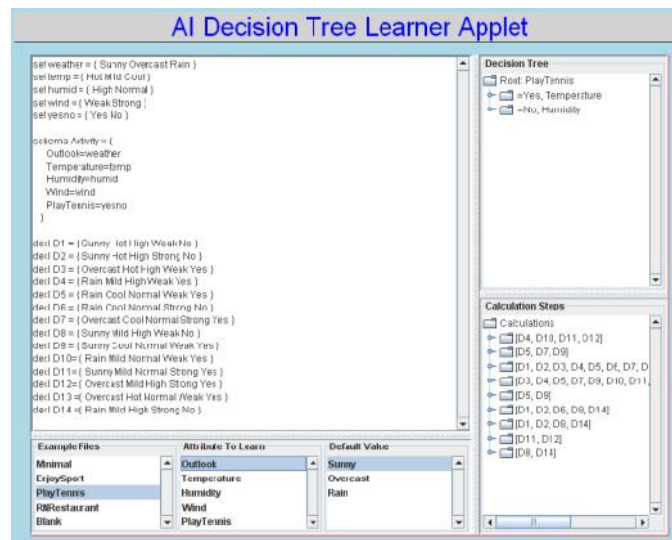
Na Univerzitetu Novog Južnog Velsa u Australiji se za potrebe lakšeg shvatanja rada stabla odlučivanja koristi odgovarajući simulator [42, 43] (grafikon 8). Mana ovog programa je što ne poseduje odgovarajući grafički izlaz, kao i što nema mogućnost kretanja korak po korak kroz sam algoritam. Na taj način, studenti samo mogu izvršiti algoritam i pregledati vrednosti entropije. I pored solidne funkcionalnosti programa, problem mogu predstavljati lakoća učenja i razumljivost korišćenja, faktora koji bi kod ovakvog simulatora trebalo da imaju izuzetno važnu ulogu.

Grafikon 8: Simulator stabla odlučivanja - Univerzitet Novog Južnog Velsa.



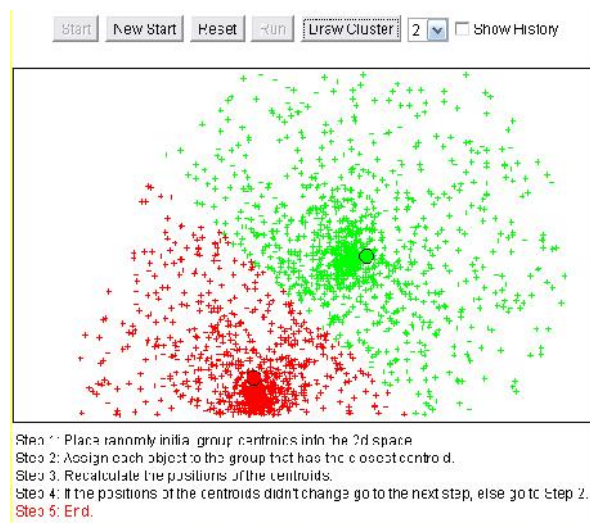
Na Imperijal Koledžu u Londonu se takođe koristi program za demonstraciju rada stabla odlučivanja [44, 45] (grafikon 9). I u ovom slučaju se javlja isti problem kao i u prethodnom – ne postoji odgovarajući grafički interfejs za prezentovanje, već je sve svedeno na tekstualni ispis. Takođe, ni ovde ne postoji mogućnost postupnog praćenja rada algoritma. Iako je program po pitanju funkcionalnosti dobro urađen, problem predstavlja lakoća učenja i pamćenja.

Grafikon 9: Simulator stabla pretraživanja - Imperijal Koledž.



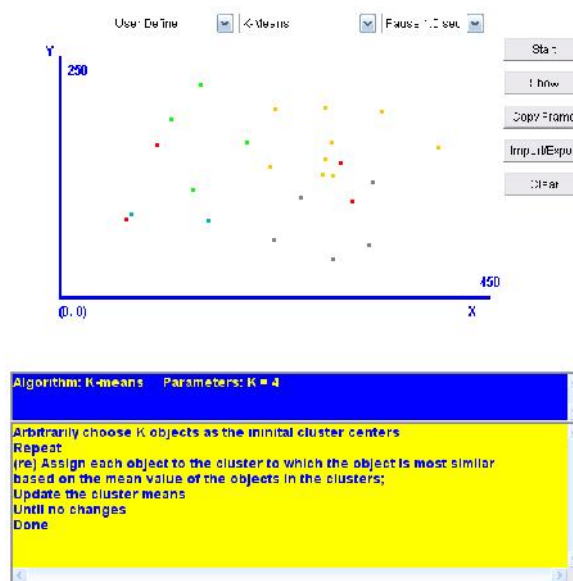
Na Tehničkom univerzitetu Braunšvajg u Nemačkoj postoji simulator koji se koristi za razumevanje klasterovanja [46] (grafikon 10). Ovaj simulator poseduje mogućnost grafičkog prikaza, ali problem je u nemogućnosti postepenog kretanja kroz algoritam. I pored toga što se ispisuje koji se korak algoritma trenutno izvršava, ne postoji ispis dodatnih informacija, kao što je broj trenutnog koraka ili broj tačaka koje su u pomerene u određenom koraku. Program ima potrebnu funkcionalnost i lakoću učenja, ali problem predstavlja razumljivost korišćenja zbog nemogućnosti kretanja kroz algoritam.

Grafikon 10: Simulator klasterovanja - Tehnički univerzitet Braunšvajg.



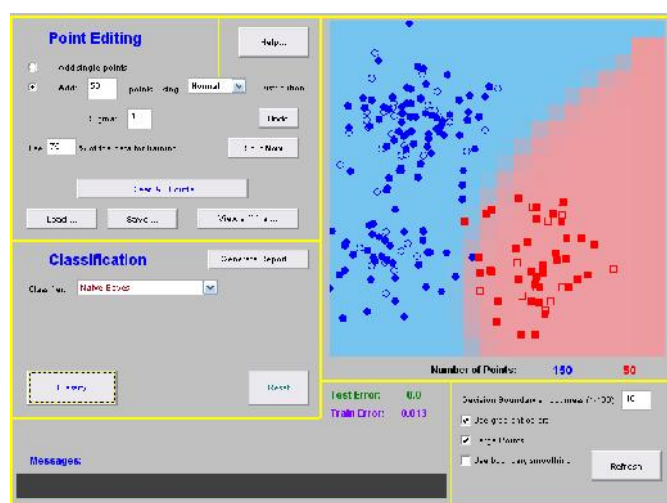
Na Univerzitetu Alberta u Kanadi, u okviru departmana za računarske nauke, koristi se program za simulaciju klasterovanja [47, 48] (grafikon 11). Program poseduje svedene mogućnosti za grafički prikaz, a problem može predstavljati i to što ne postoji grafička predstava centroida. I u ovom slučaju ne postoji mogućnost postepenog kretanja kroz algoritam, kao ni prikaz poruka o tome šta algoritam trenutno radi. Po pitanju faktora korisnosti, u ovom slučaju, situacija je identična kao i u prethodnom primeru.

Grafikon 11: Simulator klasterovanja - Univerzitet Alberta.



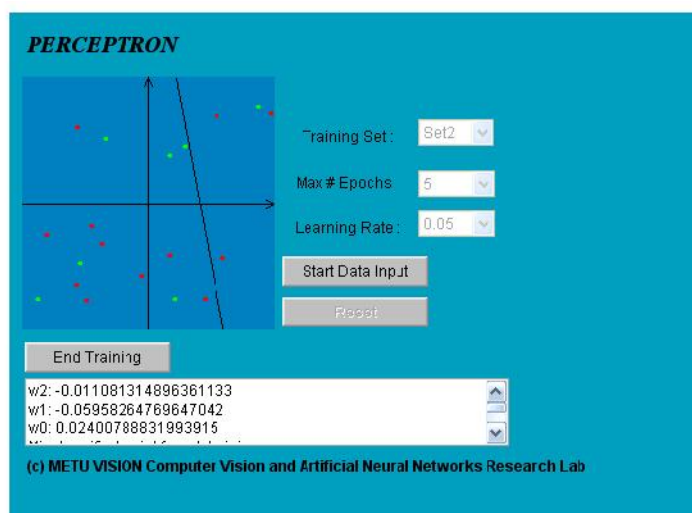
Na Tehnionu (*Technion*) – izraelskom institutu tehnologija, na smeru za računarske nauke postoji simulator klasifikacionih algoritama, među kojima je i Naive Bayes [49, 50] (grafikon 12). Ulazni podaci su predstavljeni tačkama u koordinatnom sistemu koji je predstavljen i grafički. Kao i kod svih prethodno prikazanih programa, ni ovde ne postoji mogućnost kretanja kroz algoritam. Takođe, ne postoji nikakav tekstualni izlaz pomoću koga bi koraci koje algoritam pravi bili detaljnije objašnjeni. Iako je program po funkcionalnostima prilično bogat, problem može predstavljati organizacija komandi i polja za unos parametara. To je značajno umanjilo njegovu lakoću učenja i lično zadovoljstvo korisnika.

Grafikon 12: Simulator klasifikacionih algoritama - Tehnion.



Na Blisko-istočnom tehničkom univerzitetu u Turskoj koristi se program za demonstraciju rada perceptrona [51, 52] (grafikon 13). Interfejs je prilično sveden, a kontrola rada algoritma ne postoji. Postoji dijalog za poruke, ali se ispisuju samo osnovni koraci. Program poseduje dobru lakoću učenja i pamćenja, ali problem predstavlja razumljivost korišćenja, zbog prilično jednostavnog ispisa parametara izvršavanja algoritma, kao i nedostatka mogućnosti kretanja korak po korak.

Grafikon 13: Simulator perceptrona - Blisko-istočni tehnički univerzitet.



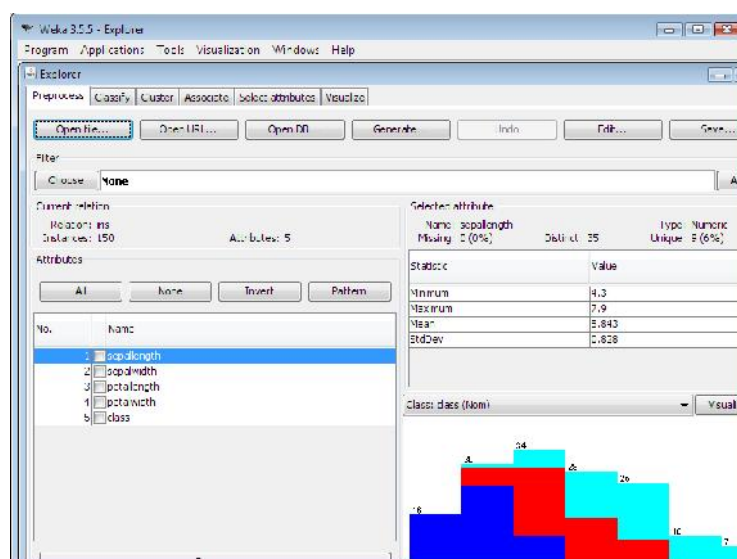
Na Univerzitetu Delft u Holandiji razvijen je pojednostavljeni simulator fudbala, a koristi se u edukacione svrhe na uvodnom kursu iz oblasti veštačke inteligencije na diplomskim studijama i služi za predstavljanje osnovnih koncepata (grafikon 14). [53]

Grafikon 14: Pojednostavljeni simulator fudbala – Univerzitet Delft.



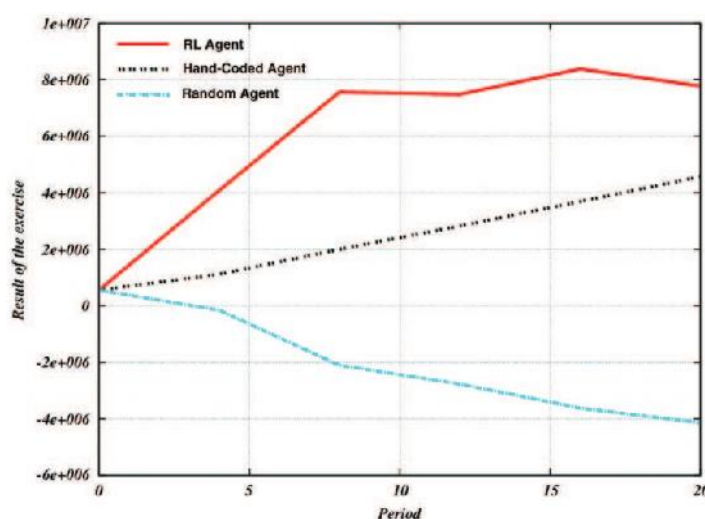
WEKA (*Waikato Environment for Knowledge Analysis*) [55] predstavlja alat za učenje algoritama mašinskog učenja kroz razne probleme iz realnog sveta (grafikon 15). Sadrži kolekciju algoritama mašinskog učenja koji se koriste za potrebe *data mining*-a [55]. Ovi algoritmi se mogu direktno primeniti na ulazni skup podataka ili se mogu pozvati iz sopstvenog Java programskog koda. Weka poseduje alate za pred-procesiranje podataka, klasifikaciju, regresiju, klasterovanje, pravila asocijacije i vizuelizaciju [56]. Simulator poseduje veliki skup mogućnosti, ali zato može biti i prilično komplikovan za korišćenje. Potrebno je određeno vreme da bi se korisnici privikli na interfejs, kao i mogućnosti programa.

Grafikon 15: Simulator WEKA.



SIMBA (SIMulator for Business Administration) [57, 58] je novi simulator razvijen kao veb platforma za rad u učionici ili učenje na daljinu (grafikon 16). Može se koristiti i kao alat za razvoj i evaluaciju inteligentnih agenata. Simulator se dosta oslanja na poslovanje preduzeća, a podstiče i takmičarski duh, budući da se nekoliko kompanije međusobno mogu takmičiti. Pristup orijentisan ka poslovanju može predstavljati problem ovog simulatora sa aspekta mašinskog učenja, budući da ono ipak nije primarna tema u simulatoru.

Grafikon 16: Simulator SIMBA.

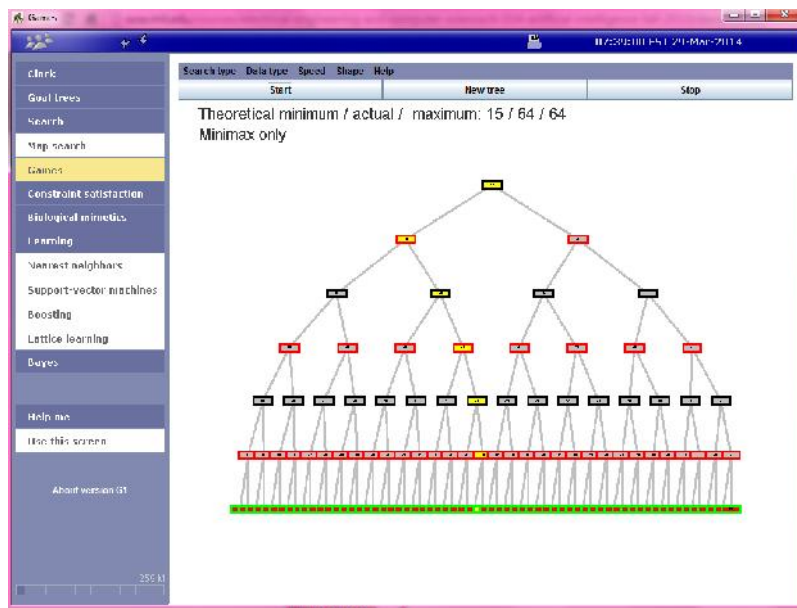


3.2.2. Postojeći simulatori algoritama pretraživanja

MIT Open CourseWare [59] koristi *Java Web Start* kao osnovu i predstavlja simulator koji se koristi na MIT-u za potrebe demonstriranja inteligentnih algoritama (grafikon 17). Obuhvaćeni su najpoznatiji algoritmi pretrage (po dubini, po širini, planinarenje, pretraga po snopu, grananje sa ograničavanjem, A*), igre koje koriste *mini-max* pretragu i *alpha-beta* odsecanje, genetski algoritmi, zadovoljenje ograničenja itd.

Iako ovaj simulator obuhvata veliki skup mogućnosti, probleme u njegovom korišćenju može stvoriti činjenica da se ne može upravljati algoritmom na precizan način brojem koraka koji odgovaraju korisniku, kao ni vratiti na neki prethodni korak. Takođe, sistem u toku rada ne daje detaljnija objašnjenja o tome šta algoritam radi u tom trenutku.

Grafikon 17: MIT Open CourseWare simulator.

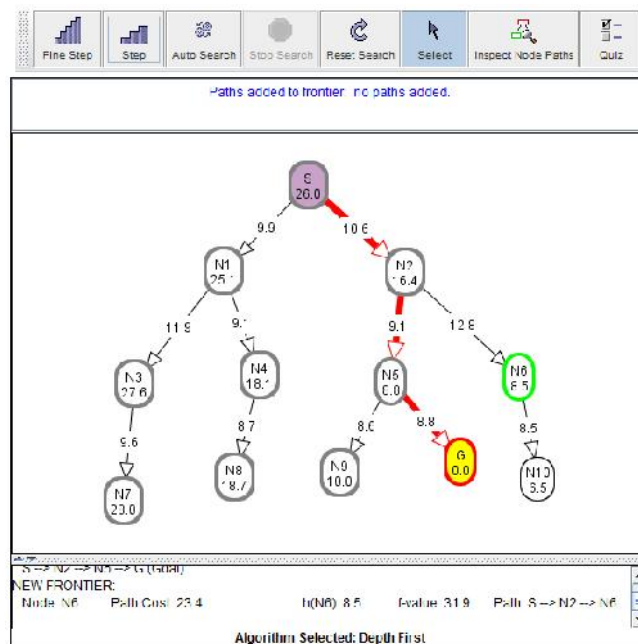


AI Space simulator (grafikon 18) [60] se koristi za učenje i istraživanje koncepata veštačke inteligencije [61]. Razvijen je na *University of British Columbia*. Obuhvata veliki broj algoritama i tehnika: grafovske pretrage, stabla odlučivanja, neuronske i Bajesove mreže itd.

I pored velike raznolikosti obuhvaćenih tehnika, uloženi su veliki napor i trud da interfejs između svih modula bude ujednačen, kako bi se korisnici što brže prilagođavali. Za svaki od njih postoje dva načina rada - *Create* mod i *Solve* mod - korisnik u svakom trenutku može izabrati

u kom želi da radi. *Create* mod obezbeđuje funkcije za pravljenje reprezentacije problema. *Solve* mod obezbeđuje funkcije za rešavanje i/ili testiranje problema. Kroz rešavanje problema se može proći odjednom, ili postepeno korak po korak. Jedini potencijalni nedostatak može predstavljati delimično konfuzan način ispisivanja poruka koje prate rad algoritama, kao i grafički aspekt izgleda simulatora.

Grafikon 18: AI Space simulator.

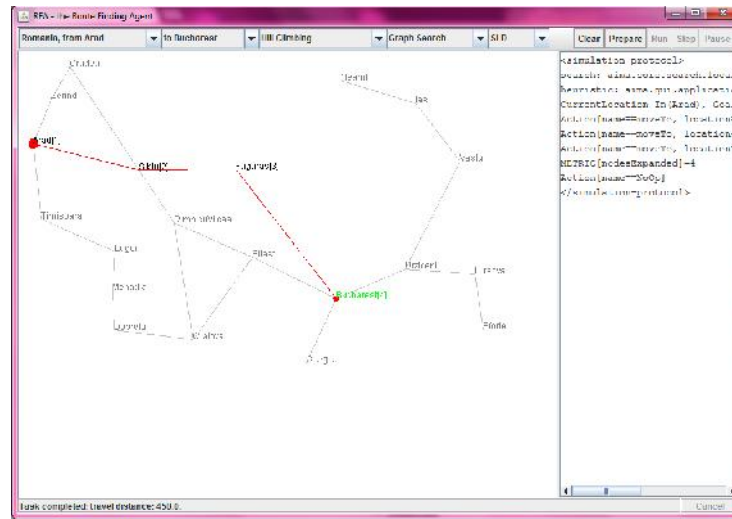


Simulator AIMA3e (*Artificial Intelligence - A Modern Approach 3rd Edition*) [62] predstavlja dopunu udžbenika *Veštačka inteligencija* koji su napisali *Stuart Russell* i *Peter Norvig* [70]. Najvažniji koncepti su predstavljeni pomoću simulatora, kako bi se olakšalo njihovo razumevanje. Obradeni su rad sa agentima, algoritmi pretraživanja (uključujući sve one koji se obrađuju i u ovom radu), korišćenje primera *8-puzzle* i *n-kraljica*, *mini-max* i *alpha-beta* algoritama, problemi bojenja mape itd.

Izgled dela simulatora koji je zadužen za pretraživanje je prikazan na grafikonu 19. Organizovan je izuzetno pregledno, i omogućava veoma lako upoznavanje i korišćenje. Opcije su pregledne, ali malobrojne. Nije moguće učitati sopstveni skup podataka, već se

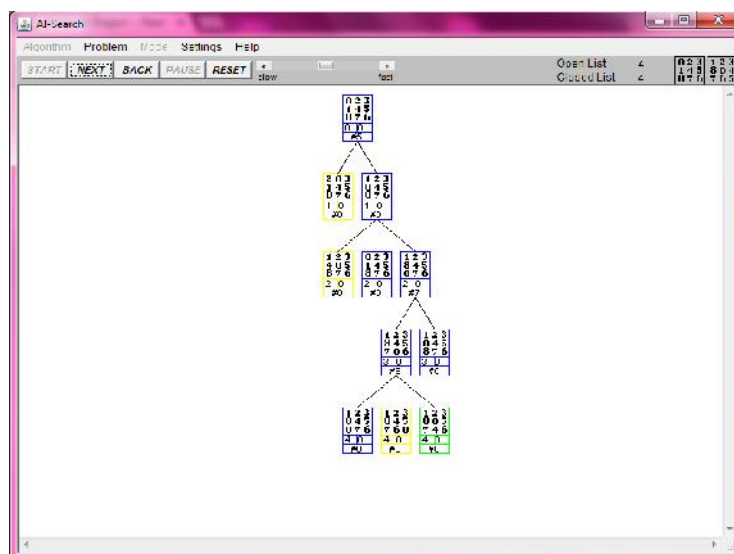
Softverski simulator za edukaciju u oblasti inteligentnih sistema

Grafikon 19: Aima3e.



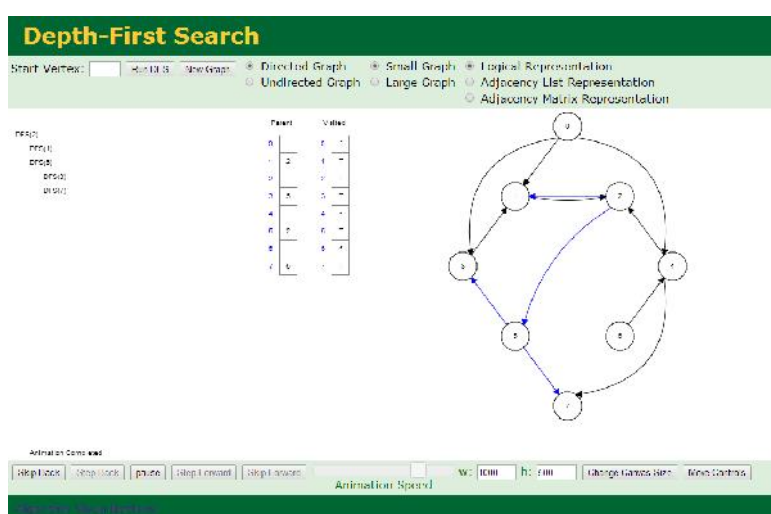
Simulator *AI-Search* [63] razvijen je RMIT Univerzitetu u Australiji (grafikon 20). Na primeru *8-puzzle* se demonstriraju najpoznatiji algoritmi pretrage kao što su pretraga po dubini i širini, A^* , pohlepni algoritmi ili iterativno produbljivanje. U simulatoru je moguće prolaziti kroz problem korak po korak. Zanimljivo je što je za demonstraciju pretraživanja upotrebljena igra *8-puzzle*. Problem predstavlja što je sistem u upotrebi oko 15 godina, i što za današnje pojmove ima prilično zastareo interfejs koji može da bude problematičan za korišćenje i razumevanje.

Grafikon 20: AI-Search.



Data Structure Visualizations predstavlja zanimljivu kolekciju simulatora koja je razvijena na *University of San Francisco* [64]. Obuhvaćeni su razni grafovski algoritmi (pretraga u širinu i dubinu), algoritmi sortiranja i indeksiranja itd. Zanimljivo je da je ceo sistem napisan pomoću *JavaScript*-a što mu omogućuje da radi na velikom broju uređaja i bez instaliranja bilo kakvih dodatnih komponenti. Interfejs je pregledno organizovan i ima lako dostupne opcije (grafikon 21).

Grafikon 21: Data Structure Visualizations.



3.3. Analiza izabranih algoritama mašinskog učenja

Kao što je prikazano u prethodnim sekcijama, u oblasti inteligentnih sistema postoji mnoštvo algoritama, a svaki je predviđen za rešavanje određenih problema. U okviru ovog rada, cilj je bio da se izaberu oni algoritmi koji su najpogodniji za što širi spektar upotrebe i koji bi studentima mogli da budu najkorisniji u izučavanju ove oblasti. Pri izboru algoritama vođeno je računa i o tome šta preporučuju strukovne organizacije, kao i šta se izučava na poznatim svetskim univerzitetima.

Na osnovu svega do sada izloženog, odlučeno je da se u ovom radu obrade stabla odlučivanja, klasterovanje, Naive Bayes, kao i perceptroni – osnovne gradivne komponente neuronskih mreža.

U narednim sekcijama, svaki od izabranih algoritama je detaljnije objašnjen. Prikazane su njihove mogućnosti i način rada, kao i najvažnije prednosti svakog od njih.

3.3.1. Stabla odlučivanja

Stabla odlučivanja se koriste kao vizuelni i analitički alati za pomoć odlučivanju, a do početka ozbiljnijeg razvoja računarskih tehnologija tradicionalno su pravljena ručno. Naravno, u slučaju složenih skupova podataka, to je bio izuzetno složen, a ponekad i potpuno neizvodljiv posao.

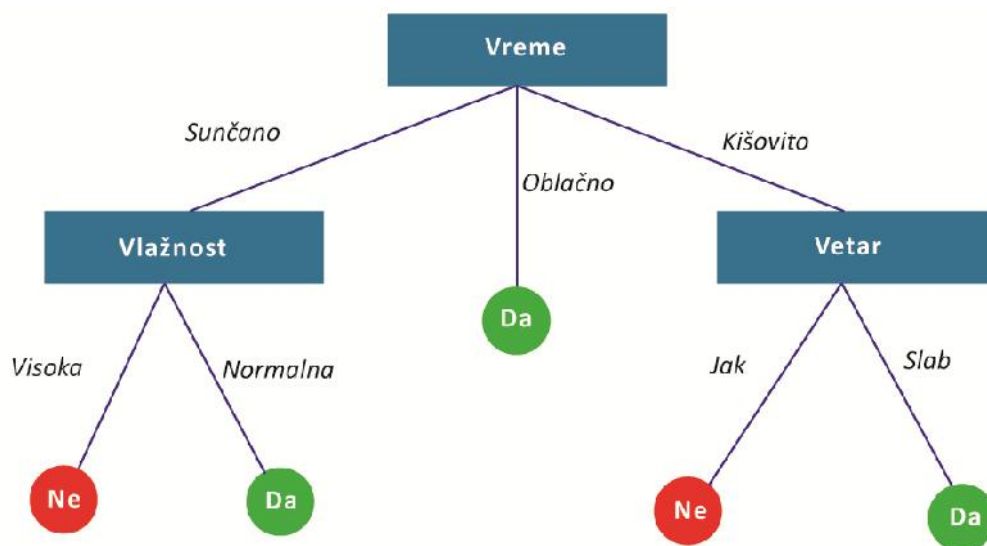
Za potrebe pravljenja stabala odlučivanja, u okviru ovog rada, upotrebljen je algoritam *ID3 (Interactive Dichotomizer 3)*, koji je razvio i predstavio *Ross Quinlan* [65]. *ID3* spada u najpoznatije u najčešće korišćene algoritme stabala odlučivanja. [66] Ovaj algoritam je u upotrebi u velikom broju ekspertskih sistema. Zasniva se na informacionoj teoriji i pokušajima minimizovanja ukupnog broja poređenja [67]. Karakterističan je i po tome što generiše plitka i široka stabla, u kojima se do rešenja stiže relativno brzo [68].

Na osnovu skupa primera problema generiše se stablo odlučivanja, pri čemu se atributi smeštaju u čvorove stabla na takav način da se minimizira pretraživanje u svrhu nalaženja rešenja. Postoji nekoliko važnih odlika ovog algoritma [69]:

- *Izbor najvažnijeg atributa:* Da bi bilo napravljeno ispravno stablo, ID3 mora biti u stanju da prepozna najvažniji atribut koji će postaviti u sam koren. Taj atribut bi preostale elemente trebao da razdvoji u što boljoj meri. Što je atribut važniji, biće postavljen bliže korenu stabla.
- *Nedostatak podataka:* Postoji mogućnost da algoritam ne bude u stanju da nađe odgovarajuće rešenje. To se događa u slučajevima kada ne postoje primeri na osnovu kojih je moguće doneti odluku. To je obično signal da početni skup primera nije dovoljan i da ga je potrebno proširiti.
- *Isključivanje nevažnih faktora:* Veoma važna odlika ID3 algoritma je mogućnost proglašavanja određenog faktora nebitnim za pravljenje stabla odlučivanja. U tom slučaju, taj faktor neće biti uključen u krajnje stablo odlučivanja.

Grafikon 22 prikazuje kako bi moglo da izgleda stablo odlučivanja za slučaj kada se određuje da li će se odigrati teniski meč.

Grafikon 22: Prikaz stabla odlučivanja.



Prilagođeno prema: Mitchell T., *Machine Learning*, McGraw Hill, Boston, 1997, str. 53.

Velika prednosti stabala odlučivanja je visok stepen grafičke preglednosti - jasno se vidi da će meč biti odigran samo ako je vreme sunčano i vlažnost normalna ili u slučaju ako pada kiša, ali je vetar slab.

Prvi korak u radu algoritma je izbor najdiskriminativnijeg atributa koji obezbeđuje što bolju klasifikacije primera. Idealno, ovaj atribut deli ulazni skup na vrednosti koje su ili samo pozitivne ili samo negativne. Naravno, takvu podelu nije uvek moguće izvesti. Da bi se odredilo koji je atribut je dobar a koji nije, uvodi se nova mera koja se naziva *entropija*.

Entropija predstavlja meru neizvesnosti neke slučajne promenljive. Na primer, kod novčića koji na obe strane ima pismo entropija će imati vrednost 0. To znači da uopšte ne postoji neizvesnost jer će uvek pasti pismo, koliko god puta novčić bio bacan. Kod ispravnog novčića, verovatnoća da padne pismo ili glava je u potpunosti jednaka – entropija u ovom slučaju ima vrednost 1 - dakle u pitanju je potpuno neuređeno stanje u okviru koga se ne može sa sigurnošću predvideti rezultat. Ako pogledamo novčić kod kog u 99% slučajeva pada pismo – ukoliko izaberemo pismo, napravićemo grešku u samo 1% slučajeva. Tada će entropija imati vrednost koja je bliska nuli, ali je i dalje pozitivna. [70]

Entropija se prema Šenonu definiše kao [71]:

$$Entropy(S) = \sum_{i=1}^c -p_i \log_2 p_i ,$$

gde je S skup ulaznih podataka, a p_i proporcija skupa S koja pripada klasi i . [72]

Na prethodnom primeru sa bacanjem ispravnog novčića možemo proveriti vrednost entropije:

$$Entropy(ucnpravan) = -(\frac{1}{2} \log_2 \frac{1}{2} + \frac{1}{2} \log_2 \frac{1}{2}) = 1.$$

Na ovaj način su potvrđene prethodne tvrdnje kada je navedeno da će vrednost entropije ispravnog novčića biti 1.

Da bi bilo moguće odrediti koji je atribut najdiskriminativniji, potrebno je izračunati informacionu dobit (*gain*). Informaciona dobit predstavlja očekivano smanjenje entropije, a računa se na sledeći način:

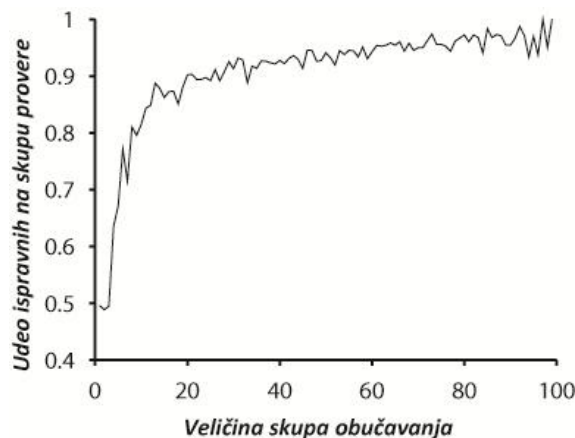
$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v),$$

Gde je A atribut, $values(A)$ je skup mogućih vrednosti atributa A , a S_v je podskup od S za koji atribut A ima vrednost v [73]. Potrebno je zapaziti da je $entropy(S)$ entropija originalne kolekcije S , dok drugi deo jednačine predstavlja entropiju nakon što se S podeli korišćenjem atributa A .

Informaciona dobit je mera koju koristi ID3 algoritam kako bi izabrao najbolji atribut prilikom pravljenja stabla odlučivanja [74]. Za pravljenje stabla, biće izabran onaj atribut koji ima najveću informacionu dobit. Svrha ovakvog pristupa jeste da se zapravo naprave mala stabla odlučivanja kako bi se što pre, uz što manji broj podela ulaznog skupa, došlo do krajnje odluke.

Preciznost stabla odlučivanja se može proceniti pomoću krive obučavanja koja je data na grafikonu 23. Kriva obučavanja pokazuje na koji način se menja preciznost algoritma u odnosu na ulazne podatke.

Grafikon 23: Kriva obučavanja za stablo odlučivanja.

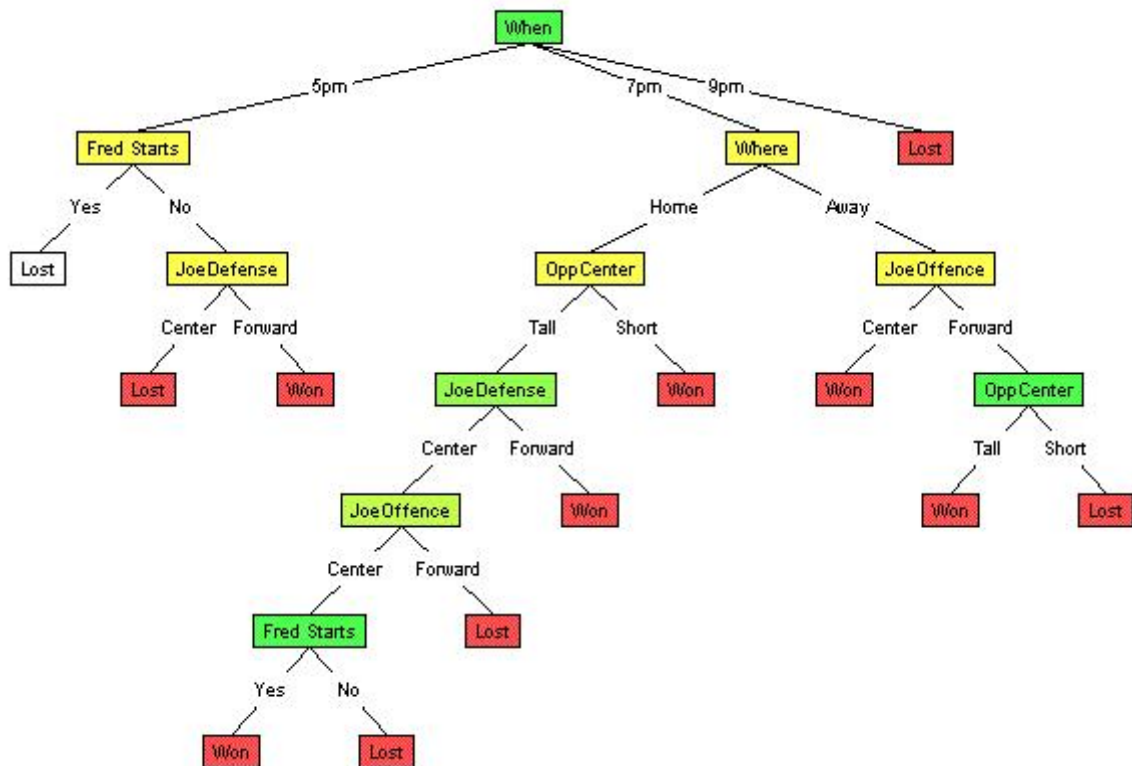


Izvor: Russell S., Norvig P., Veštačka inteligencija – savremeni pristup, CET, Beograd, 2011, str. 763.

Navedeni grafikon prikazuje da sa porastom veličine skupa obučavanja, raste i udeo ispravnih odluka. To nedvosmisleno pokazuje koliko je bitno obezbediti što veći broj primera za obučavanje kako bi stablo odlučivanja donosilo što ispravnije odluke. Jasno se vidi da preciznost dostiže vrednost od 95%, a verovatno bi i nastavila da raste, kada bi se nastavio unos ulaznih podataka.

U krajnjim čvorovima stabla (listovima) nalazi se odgovor na određeno pitanje. Bolje je da se odgovor nalazi što bliže vrhu stabla (korenu) jer će tada biti precizniji. Što je više čvorova potrebno preći da bi se stiglo do odgovora, on postaje neprecizniji. Zanimljiv primer je prikazan u okviru grafikona 24. Na njemu je predstavljeno predviđanje da li će tim pobediti ili izgubiti na određenoj utakmici.

Grafikon 24: Veza između visine stabla i preciznosti odgovora.



Izvor: Alxploratorium, University of Alberta [75]

Ukoliko se pogleda leva strana stabla, videće se da ukoliko se utakmica igra u 5 časova poslepodne i *Fred* započne igru, velika verovatnoća je da će meč biti izgubljen - odgovor se nalazi veoma blizu korenu stabla. Ili ukoliko se bude igralo u 9 sati uveče, meč je gotovo

sigurno izgubljen. Može se pogledati i složenija pretpostavka: ukoliko se bude igralo u 7 sati poslepodne na domaćem terenu, protivnički igrač na centru bude visok, *Joe* iz odbrane igra ka centru i pritom ne čini prekršaje u napadu, i ako pored svega toga *Fred* započne utakmicu, može se očekivati pobjeda. Kao što se može pretpostaviti, u ovom složenom lancu događaja koji bi trebao da se desi, dovoljno je da samo jedna karika dâ nepredviđen rezultat i možemo očekivati potpuno drugačiji ishod utakmice - u ovom slučaju gubitak.

3.3.2. Klasterovanje

Klasterovanje podrazumeva deljenje podataka u grupe koje imaju određeno značenje i koje bi trebalo da odražavaju prirodnu strukturu podataka. Klasterovanje već dugo vremena ima važnu ulogu u mnogim poljima: psihologija i druge društvene nauke, biologija, statistika i mašinsko učenje. [76]

Postoji mnogo načina da se klasteri primene na praktične probleme. U ovom radu obrađuje se algoritam *k-means* koji je prvi put predložio *Stuart Lloyd* 1957. godine. Zbog toga se ovaj algoritam često naziva i *Lloyd-ov* algoritam. [77] Zasnovan je na jednostavnom zapažanju da optimalni centar treba postaviti u centroidu određenog klastera [78]. *K-means* je jedan od najstarijih i najviše korišćenih algoritama klasterovanja. [79, 80]

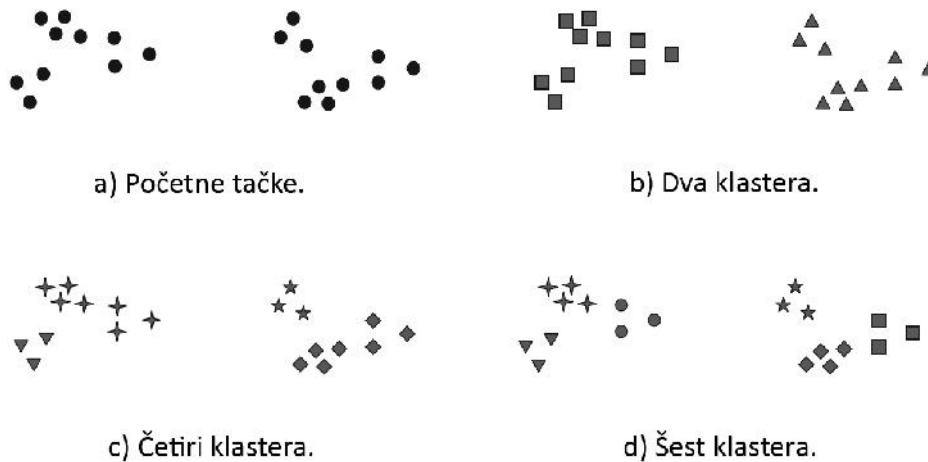
Zbog svoje jednostavnosti i fleksibilnosti, ovaj algoritam je popularan u oblasti statističke analize. Može se koristiti i u post-obradi rezultata rada nekog drugog algoritma za klasterovanje, kako bi dodatno poboljšao krajnje rešenje. To je svakako važna mogućnost koja ima svoje prednosti, ali to u određenim situacijama može uticati na brzinu rada algoritma - problem je u tome što se u svakom koraku obračunavaju najbliži susedi. [81]

Klasterovanje podrazumeva da se objekti dele u grupe prema određenom kriterijumu. Cilj je da objekti u grupi budu što sličniji, a da razlike između grupa budu što veće.

Da bi se lakše prikazalo kako se podaci mogu grupisati, na grafikonu 25 a) je prikazano 20 tačaka. One se na više različitih načina mogu smestiti u klastere - 25 b) prikazuje podelu početnih podataka u dve grupe – očigledno je da su tačke logički grupisane. Isto važi i za primere 25 c) i 25 d) gde se vidi podela na četiri i šest grupa, respektivno. Na ovim

primerima se najbolje vidi da je definicija klastera neprecizna i da je veoma zavisna od prirode podataka i željenih rezultata.

Grafikon 25: Prikaz nekoliko načina klasterovanja.



Prilagođeno prema: Tan Pang-Ning, Steinbach M., Kumar V., *Introduction to Data Mining*, Addison-Wesley, Boston, 2006, str. 491.

Algoritam *k-means* je particionalna tehnika klasterovanja koja pokušava da nađe broj klastera koje je odredio korisnik, a koji su predstavljeni svojim centroidima. Pod particionalnim klasterovanjem se podrazumeva takvo klasterovanje gde se ceo skup podataka deli u nepreklapajuće skupove. Jedan objekat se može naći samo u jednom podskupu.

Centroid (prosečna tačka) je obično aritmetička sredina grupe tačaka. U okviru klastera, centroid će veoma retko biti neka već postojeća tačka, osim ukoliko to nije izričit zahtev.

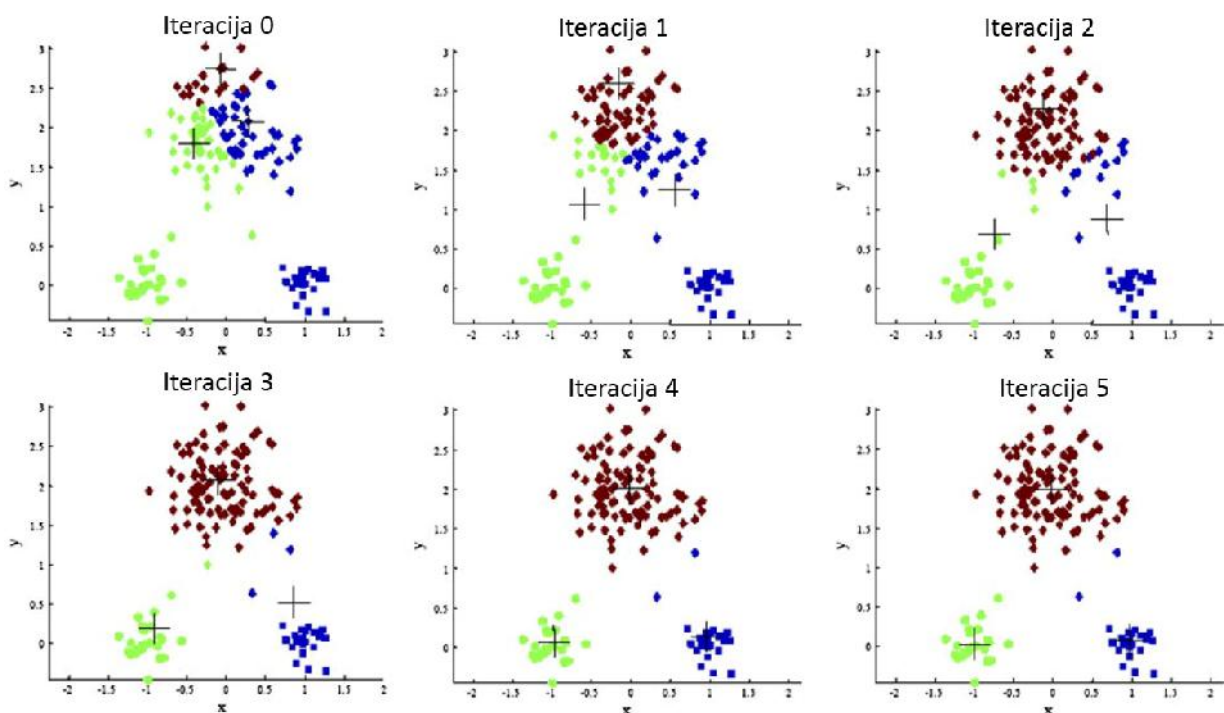
Prikazana je osnova *k-means* algoritma [82]:

1. Izabрати K tačaka kao početne centroide.
2. Ponavljati
3. Formiranje K klastera dodeljivanjem svake tačke najbližem centroidu.
4. Ponovno računanje centroida svakog klastera.
5. Dok se centroidi menjaju.

U prvom koraku, biraju se početni centroidi. Nakon što se tačke dodele centroidima, vrši se njihovo preračunavanje. Kada centroidi prestanu sa menjanjem svojih pozicija, znači da su formirane prirodne grupe objekata pa algoritam može prestati sa izvršavanjem. Budući da se većina pomeraja tačaka događa u ranijim fazama, uslov iz tačke pet se može zameniti blažom formom – npr. ponavljati samo dok više od 1% tačaka menja pozicije.

Grafikon 26 prikazuje postupak rada algoritma k-means. U prvom koraku (iteracija 0) se određuju centroidi i početni klasteri. Bitno je naglasiti da algoritam može biti osetljiv na izbor početnih centroida, pa za različite početne centroide, može dati različita krajnja rešenja iako se radi o istom skupu ulaznih vrednosti. U svakoj sledećoj iteraciji algoritma, izračunavaju se novi centroidi, a klasterima se dodeljuju nove tačke. U poslednjoj, šestoj, iteraciji, došlo se do stanja kada ne dolazi do pomeranja (ili dolazi, ali u veoma maloj meri) centroida, kao i tačaka između postojećih klastera.

Grafikon 26: Demonstracija rada algoritma k-means.



Izvor: Grupa autora, *Top 10 algorithms in data mining*, Springer, London, 2007, str. 7.

Za merenje razdaljine između tačaka prilikom računanja pozicije centroida koristi se Euklidovo rastojanje koje u osnovi koristi Pitagorinu teoremu. Ako su date tačke rastojanje između njih se računa po formuli:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}.$$

Euklidovo rastojanje je važno jer se pomoću njega minimizuje rastojanje između centroida (kao centralnih tačaka) i drugih tačaka u određenom klasteru. Ovo rastojanje uvek mora biti pozitivno.

3.3.3. Naive Bayes

Naive Bayes modeli se često, i sa dosta uspeha, koriste za klasterovanje [83] i klasifikaciju [84]. U mnogim situacijama pokazuju izuzetno dobre karakteristike, a njihova preciznost i vreme učenja se mogu meriti i sa drugim modelima, kao što su Bajesove mreže [85], čiju jednu vrstu zapravo i predstavljaju. Čini ga mešavina komponenti, gde se u okviru svake komponente promenljive smatraju međusobno uslovno nezavisnim. Uz dovoljan broj podataka, može predvideti rešenje sa velikim stepenom preciznosti.

Algoritam Naive Bayes, kao i većina sistema koji koriste rasuđivanje zasnovano na verovatnoći, imaju za temelj Bajesovo pravilo (Bajesovu teoremu). Opšti slučaj ove teoreme može se zapisati u sledećem obliku: [70]

$$P(b | a) = \frac{P(a | b)P(b)}{P(a)}.$$

Zahvaljujući Bajesovom pravilu, možemo izračunati $P(b/a)$ pomoću $P(a/b)$, $P(b)$ i $P(a)$. Iako, površno gledano, to ne izgleda naročito korisno, zapravo u praksi često postoje situacije kada se poznate ove tri vrednosti, a traži se četvrta.

Bitno je pomenuti i poreklo naziva algoritma – iako bi se iz naziva moglo zaključiti da se radi o „naivnom“ algoritmu, zapravo se radi o tome da se podrazumeva da su atributi između sebe uslovno nezavisni. Na taj način, svaki atribut doprinosi donošenju krajnje odluke u jednakoj meri i potpuno nezavisno od drugih atributa, što je efikasnije od drugih klasifikatora, gledano iz aspekta broja potrebnih operacija [86]. Zahvaljujući tome, Naive Bayes modeli u praksi mogu raditi iznenađujuće dobro.

S obzirom na navedenu pretpostavku o međusobnoj nezavisnosti atributa, verovatnoća posmatranja konjunkcije $a_1, a_2, \dots, a_n$ je proizvod verovatnoća pojedinačnih atributa [87]:

$$P(a_1, a_2, \dots, a_n | v_j) = \prod_i P(a_i | v_j)$$

Algoritam Naive Bayes koristi ovu tvrdnju kao osnovu svog rada, a ona čini osnovu njegovog funkcionisanja:

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

Vrednost v_{NB} označava izlaznu vrednost algoritma. V predstavlja konačni skup ulaznih vrednosti, a njeni pojedinačni članovi obeleženi su sa v_j . Svaki primer iz skupa obučavanja predstavljen je kao skup vrednosti atributa $(a_1, a_2, \dots, a_n)$. Bitno je naglasiti da je broj različitih $P(a_i | v_j)$ uslova, koji moraju biti procenjeni iz skupa obučavanja, jednak broju različitih vrednosti atributa pomnoženi sa brojem različitih ciljnih vrednosti. [72]

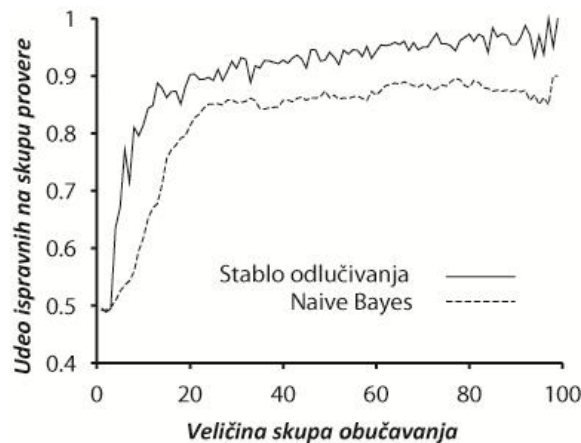
Interesantna razlika algoritma Naive Bayes u odnosu na druge algoritme mašinskog učenja (npr. stablo odlučivanja) je da ne postoji izričita pretraga prostora mogućih hipoteza. Umesto toga, hipoteza se formira bez pretraživanja, prebrojavanjem frekvencija raznih kombinacija podataka u okviru skupa obučavanja. [72]

Ovaj algoritam se pokazao izuzetno efikasnim u širokoj lepezi primena, posebno zbog svoje jednostavnosti u fazi učenja i klasifikacije [88]. Izuzetno puno se koristi i u oblasti klasifikacije teksta i slika, što je do sada potvrđeno u mnogim radovima i primerima iz prakse [89, 90]. Njegova važna karakteristika je da se dobro prilagođava i veoma velikim

problemima: sa n Bulovih atributa, postoji samo $2n+1$ parametara. Bitno je naglasiti i da nema poteškoća sa nedostajućim podacima i podacima sa šumom. [70]

Na grafikonu 27 prikazana je kriva obučavanja algoritma Naive Bayes. Kao poređenje, prikazana je i kriva obučavanja stabla odlučivanja. Vidi se da, za zadati primer, stablo obučavanja nešto bolje obučava. Naravno, za neke druge potrebe, Naive Bayes bi mogao postići bolje rezultate.

Grafikon 27: Kriva obučavanja za algoritam Naive Bayes. Kao poređenje, prikazana je i kriva obučavanja za stablo odlučivanja na istom primeru.



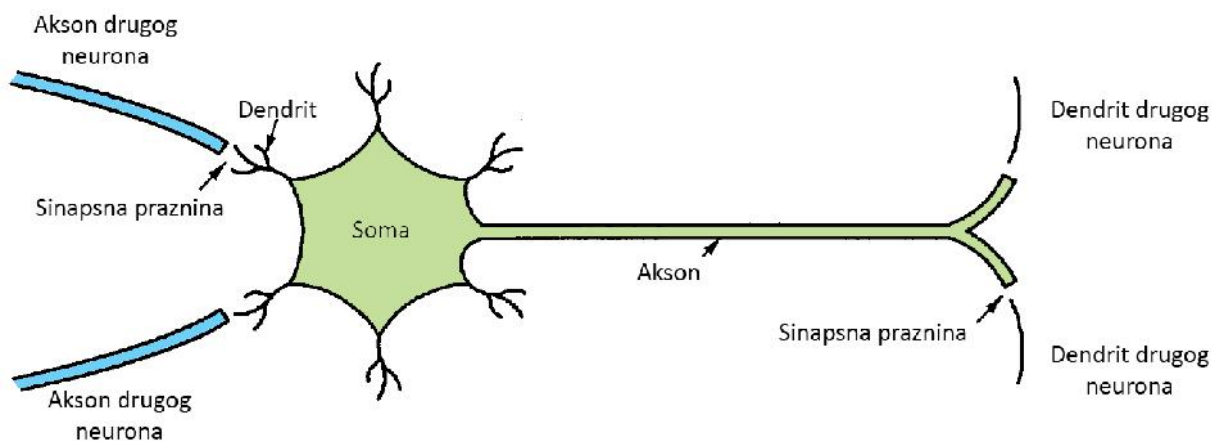
Izvor: Russell S., Norvig P., *Veštačka inteligencija – savremeni pristup*, CET, Beograd, 2011, str. 875.

3.3.4. Perceptroni

Razmišljanja o ljudskom mozgu su doprinela da se proširi domet tehničkih ideja. Među prvim radovima koji su objavljeni u oblasti veštačke inteligencije, *McCulloch* i *Pitts* početkom 1940.-ih godina objavljuju prve pokušaje pravljenja veštačkih neuronskih mreža [91]. Nakon tog perioda počinju da se razvijaju detaljniji i realističniji modeli. Danas su neuronske mreže jedan od najpopularnijih i najefikasnijih oblika sistema obučavanja i zaslužuju samostalno proučavanje. Pogodne su za rešavanje izraženo nelinearnih problema [92]. U stanju su da nauče određene nedinamične osobine sistema, a potom i da preuzmu kontrolu nad njim.

Na grafikonu 28 je šematski prikazan tipičan neuron. Svaki neuron se sastoji od tela ćelije ili some koje sadrži jezgro (nukleus). Od tela ćelije se grana određeni broj vlakana koja se nazivaju dendriti i jedno dugo vlakno koje se naziva akson. Akson se proteže prilično velikom dužinom – dužinom većom nego što prikazuje odnos na ovom grafikonu. Neuron ostvaruje spojeve sa 10^3 do 10^4 drugih neurona, a njihove veze se nazivaju sinapse [93].

Grafikon 28: Delovi nervne ćelije neurona.

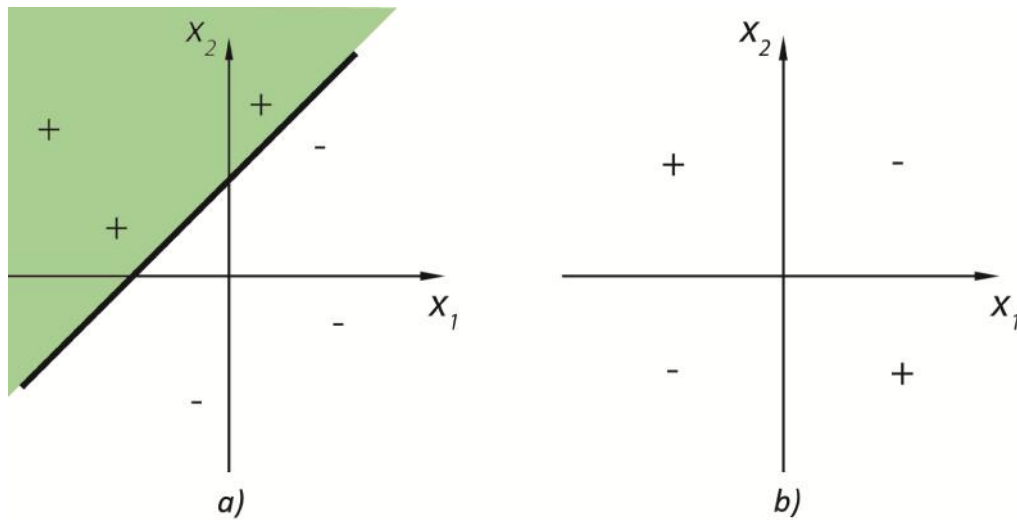


Prilagođeno prema: Fausett L., *Fundamentals of Neural Networks - Architectures Algorithms and Applications*, Prentice Hall, Upper Saddle River, 1994, str. 6.

Mreža sa svim ulazima direktno povezanim sa izlazima naziva se jednoslojna neuronska mreža ili perceptron. Od svih neuronskih mreža, perceptroni su najbolje shvaćeni i najšire korišćeni. Termin perceptron je prvi put upotrebio *Frenk Rozenblat* 1958. godine. Njegova ideja je bila da napravi funkcionalni opis načina rada pravog neurona i da ga potom implementira kao softverski algoritam. [94]

Da bi se najbolje razumela primena perceptrona, potrebno je pogledati praktičan primer. Ukoliko je dat skup tačaka koje imaju ili pozitivnu ili negativnu vrednost (svaka od njih je predstavljena sa + ili -), perceptron se može obučiti da odredi pravu koja će izvršiti podelu skupa na pozitivne i negativne vrednosti. Skup ulaznih podataka za koje je moguće odrediti pravu koja ga deli na dva skupa naziva se *linearно separabilan skup* (grafikon 29 a)). Naravno, uvek će postojati skupovi koje nije moguće homogeno podeliti (grafikon 29 b)).

Grafikon 29: Podela ulaznog skupa pomoću perceptrona.



Prilagođeno prema: Mitchell T., *Machine Learning*, McGraw Hill, Boston, 1997, str. 87.

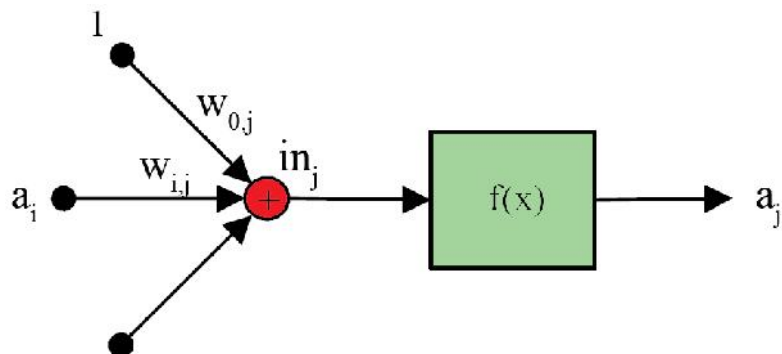
Neuronske mreže se sastoje od čvorova povezanih usmerenim vezama (grafikon 30). Veza od jedinice i do jedinice j služi za prostiranje aktivacije koja se označava sa a_i od i do j . Svaka veza ima pridruženu i numeričku težinu $w_{i,j}$. Sve jedinice imaju po jedan ulaz $a_0 = 1$ sa odgovarajućom težinom $w_{0,j}$. Svaka jedinica j prvo izračunava ponderisani zbir svojih ulaza [95, 96]:

$$in_j = \sum_{i=0}^n w_{i,j} a_i.$$

Potom se na ovaj zbir primenjuje funkcija aktivacije f da bi se dobio izlaz:

$$a_j = f(in_j) = f\left(\sum_{i=0}^n w_{i,j} a_i\right).$$

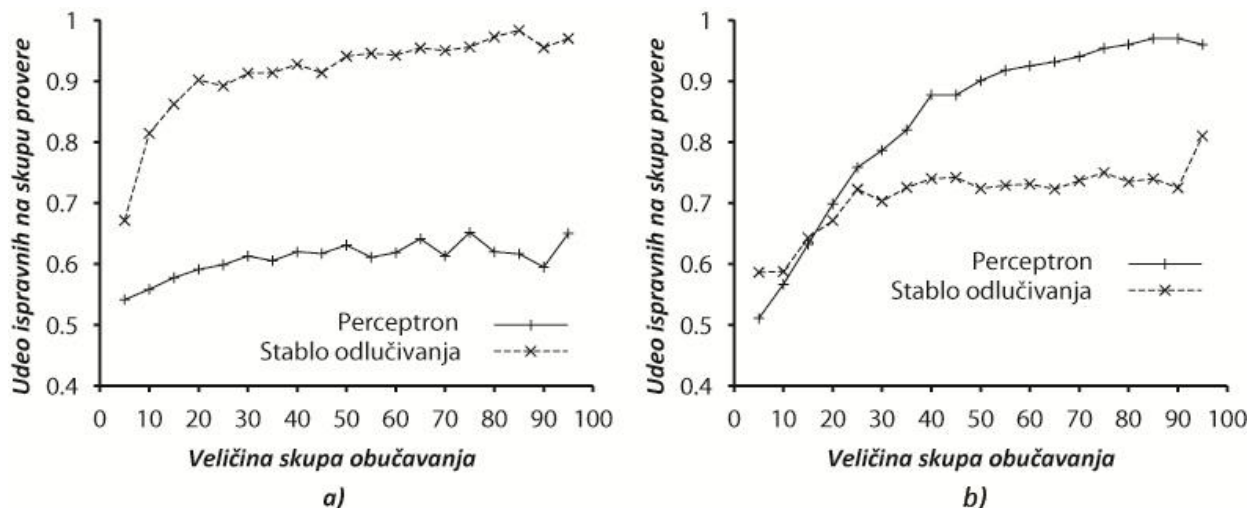
Grafikon 30: Matematički model neurona.



Prilagođeno prema: Jantzen J., *Introduction to Perceptron Networks*, Technical University of Denmark, Lyngby, 1998, str. 5.

Na grafikonu 31 je prikazana kriva obučavanja perceptrona i, radi poređenja, stabla odlučivanja. U prvom slučaju, pod a), radi se o identičnom primeru na kom je testirano stablo odlučivanja (grafikon 23). Očigledno je da u tom slučaju, perceptron znatno slabije obučava, iako su uslovi ispitivanja identični. Na istom grafikonu pod b) perceptron mnogo bolje obučava funkciju većine sa 11 ulaza.

Grafikon 31: Kriva obučavanja za perceptron. Prikazano je poređenje sa stablom odlučivanja u dve specifične situacije.



Izvor: Russell S., Norvig P., *Veštačka inteligencija – savremeni pristup*, CET, Beograd, 2011, str. 793.

3.4. Analiza izabranih algoritama pretraživanja

Pretraživanje predstavlja jednu od najčešćih aktivnosti u algoritamskim i programskim rešenjima, pa se zbog toga velika pažnja posvećuje algoritmima pretraživanja. Pretraživanje predstavlja lociranje željenog podatka u skupu podataka koji se nalaze u memoriji na osnovu određene identifikacije u cilju pristupa tom podatku. [97] Često operacije pretraživanja dominantno određuju vremensku složenost nekih algoritama, tako da je potrebno poznavati efikasnost ovih metoda, posebno ako se radi o velikim skupovima podataka.

Algoritmi pretraživanja se mogu svrstati u dve klase: algoritme globalnog (sistematskog) i algoritme lokalnog pretraživanja.

Algoritmi globalnog pretraživanja pretražuju prostor na sistematski način. Ako je taj prostor konačan, rešenje će ili biti pronađeno, ili će biti javljeno da ono ne postoji. [98] Sistematičnost se postiže tako što se putanje čuvaju u memoriji i što se beleži koje su alternative istražene u svakoj tački te putanje. Kao deo pronađenog rešenja, čuva se i putanja do tog cilja. Primeri ovakvih algoritama su pretraživanje po širini i pretraživanje po dubini.

Algoritmi lokalnog pretraživanja funkcionišu samo uz pomoć jednog trenutnog čvora (a ne pomoću većeg broja putanja) i uglavnom prelaze samo u susede tog čvora. Najčešće se putanje kojima je pretraživanje izvršeno ne čuvaju. I pored toga što ovi algoritmi nisu sistematični, pogodni su jer koriste veoma male količine memorije. Takođe, mogu naći rešenje u velikim i neprekidnim grafovima za koje sistematski algoritmi nisu pogodni. Ovakvi algoritmi su pogodni za situacije u kojima je bitno jedino konačno rešenje, a ne putanja do tog rešenja - na primer u projektovanju integrisanih kola ili telekomunikacionih mreža. Tipičan predstavnik ove klase algoritama je planinarenje.

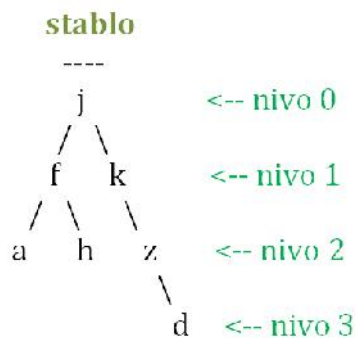
Postoji veliki broj ovakvih algoritama, a za potrebe razvoja novog simulatora, izabrana su četiri najpoznatija: pretraživanje po širini i pretraživanje po dubini, planinarenje i A*. Svaki od ovih algoritama ima određene prednosti i mane, a njihova uspešnost u radu zavisi i od ulaznog skupa podataka koji se koristi. U određenim situacijama, svaki od njih se može pokazati i kao dobro rešenje, a nekada i kao ne tako dobar izbor.

U narednim sekcijama, izabrani algoritmi su bliže predstavljeni, a definisane su i njihove prednosti i mane, kao i objašnjen njihov način rada.

3.4.1. Pretraga po širini

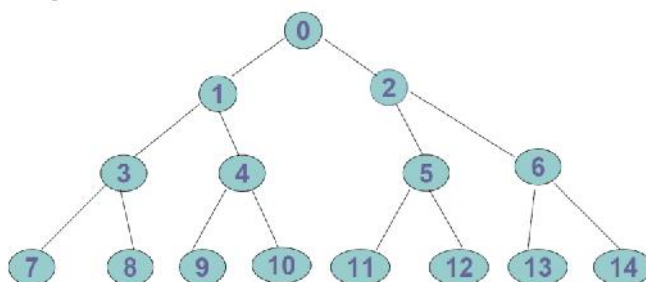
Pretraga po širini (*Breadth first search - BFS*) je izuzetno popularna tehnika i relativno laka za razumevanje i predstavlja fundamentalnu tehniku pretrage [99]. Danas se izuzetno puno koristi i za pretraživanje velikih mreža (kao što su društvene mreže) [100]. *BFS* se može primeniti i na usmerene i na neusmerene grafove, a predstavlja način pronalaženja svih čvorova, koji su dostupni iz korenog čvora *s* od koga se počinje pretraga. Osnovna ideja algoritma je sledeća: *poslati talas iz korenog čvora s*. Talas prvo obuhvata sve čvorove koji se nalaze udaljeni jednu granu od korena (prvi nivo - elementi *f* i *k*, grafikon 32), potom obuhvata sve koji su udaljeni dve grane od korena (drugi nivo - elementi *a*, *h* i *z*), itd. Kako bi se talas kontrolisao, koristi se FIFO lista – određeni čvor će se nalaziti u okviru liste ako i samo ako je talas naišao na taj čvor, ali još uvek nije prešao sa njega na njegove sledbenike. [101, 102]

Grafikon 32: Pretraga stabla pomoću *BFS* algoritma.



Ovakav obilazak nivo po nivo se naziva pretraga po širini zato što se prvo istražuje ceo jedan nivo po širini, pre nego što se pređe na sledeći [103]. Grafikon 33 predstavlja primer obilaska stabla po širini - redosled obilaska svakog elementa predstavljen je odgovarajućim rednim brojem.

Grafikon 33: Redosled obilaska stabla pomoću BFS algoritma.



Struktura rada algoritma je organizovana na sledeći način [69]:

1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste ciljni čvor .
 - 2.1. Ako je prvi element ciljni čvor, ne raditi ništa.
 - 2.2. Ako prvi element nije ciljni čvor, ukloniti prvi element iz liste i dodati njegove sledbenike iz stabla pretrage (ako ih ima) na kraj liste.
3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

Ova strategija, teorijski, može da bude korisna za rešavanje mnogih problema. Na prvi pogled, ona izgleda da nema nedostataka jer će rešenje, ako je prostor stanja konačan, uvek biti nađeno, ukoliko postoji. Osim toga, ono daje i najkraću putanju. Na žalost, broj čvorova koji mora da se ispita raste eksponencijalno sa nivoom. Stoga i vreme pretraživanja raste eksponencijalno, te strategija garantuje rešenje samo ako se raspolaže sa neograničenim vremenom i memorijom.

Vremenska kompleksnost pretrage po širini prilikom primene na grafove (usmerene ili neusmerene) sa n čvorova i m grana je ista kao i kod pretrage po dubini – $\Theta(n+m)$. Ukoliko je graf povezan ili je $m \geq n$, tada je vremensko kompleksnost samo $\Theta(m)$. [104]

Pretraga u širinu je posebno pogodna za korišćenje sa beskonačnim ili izuzetno velikim grafovima. Tada drugi algoritmi (kao što je pretraga u dubinu) ne mogu biti primenjeni jer bi

to dovelo do beskonačne rekurzije ili bi dovelo do izuzetno velike neefikasnosti budući da bi se nepotrebno ispitivali veliki delovi ogromnog grafa. Pretraga u širinu bi omogućila da se takvi grafovi delimično ispituju. Druga pogodna primena pretrage u širinu bi bilo rešenje problema nalaženja dužine najkraćeg puta u težinskim grafovima od datog čvora. [105]

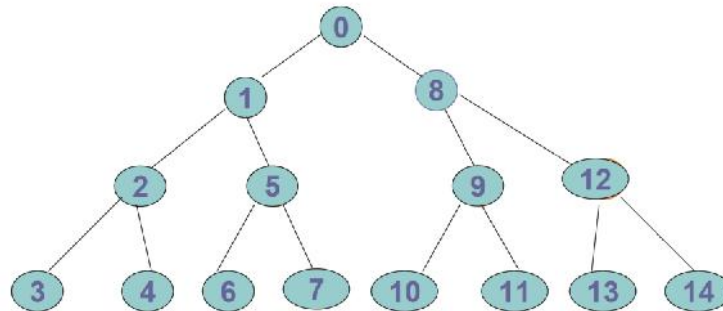
3.4.2. Pretraga po dubini

Pretraga po dubini (*Depth first search - DFS*) spada u *backtracking* tehnike i predstavlja značajan algoritam koji se puno koristi za nalaženja rešenja problema u kombinatorici i veštačkoj inteligenciji. Na njegovoj osnovi je bazirana čitava klasa drugih algoritama pretrage. [106] Praktično, *DFS* se može koristiti kao skelet za građenje efikasnih grafovskih algoritama tako što se u njegovu strukturu dodaju delovi specifični za algoritam koji se nadograđuje [107].

Kao prvi korak pretrage po dubini, svi čvorovi se označavaju kao neposećeni. Potom se izabere jedan od njih i istražuje se grana koja vodi do novog čvora. Na ovaj način se nastavlja sve dok se ne dođe do čvora preko kog se više ne može doći do sledećeg koji već nije posećen. Od ovog koraka se nastavlja unazad, sve dok se ne dođe do čvora preko koga se može doći do novog koji do sada nije posećen. Na taj način, pretraga se nastavlja dokle god postoje veze koje nisu posećene i preko kojih se može doći do novih čvorova, ili dok se ne pronađe traženo rešenje. Svi posećeni čvorovi koji potencijalno mogu imati neposećene veze čuvaju se na *steku (LIFO lista)*. Ovakav algoritam je relativno lako isprogramirati na iterativni ili rekurzivni način, pod uslovom da je obezbeđena dobra računarska reprezentacija grafa.

Grafikon 34 predstavlja primer obilaska stabla po dubini - redosled obilaska svakog elementa predstavljen je odgovarajućim rednim brojem.

Grafikon 34: Redosled obilaska stabla pomoću DFS algoritma.

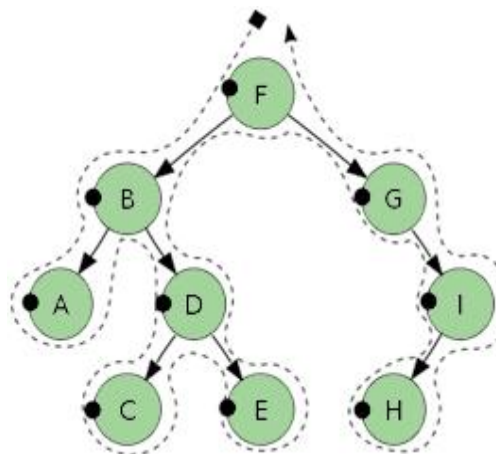


Sam algoritam ima sledeću strukturu:

1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste ciljni čvor.
 - 2.1. Ako je prvi element ciljni čvor, ne raditi ništa.
 - 2.2. Ako prvi element nije ciljni čvor, ukloniti prvi element iz liste i dodati njegove sledbenike iz stabla pretrage (ako ih ima) na početak liste.
3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

DFS pretraga grafa je najbližnja *preorder* obilasku stabala - *preorder* obilazi svaki čvor pre nego što nastavi sa obilaženjem njegove dece. Grafikon 35 predstavlja obilazak binarnog stabla na *preorder* način.

Grafikon 35: Obilazak stabla na *preorder* način.



Prilagođeno prema: [wikipedia.org](https://www.wikipedia.org)

U slučaju da se pretražuje grafovska struktura, može se desiti da se isti čvorovi obilaze po nekoliko puta, što može predstavljati veliki problem, zbog čega složenost pretrage može rasti eksponencijalno [108]. Može se desiti i da algoritam pretraživanja bude zauzet na nekom nivou (zbog postojanja beskonačne putanje) i da se tada rešenje uopšte ne nađe, iako ono postoji.

Za razliku od pretrage po širini, pretraga po dubini zahteva znatno manje memorije, ali zato ne garantuje nalaženje najkraće putanje [109]. Značajna prednost je i to što će problemi koji imaju rešenja na dubljim nivoima biti pre rešeni.

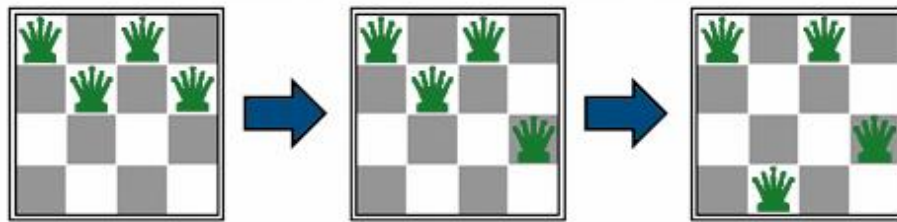
3.4.3. Planinarenje

U odnosu na algoritme pretrage u širinu i dubinu koji su prikazani, planinarenje (*Hill-climbing*) predstavlja potpuno drugačiji pristup problemu. Algoritam počinje od slučajne pretpostavke rešenja. [110] Koristi se jednostavna petlja koja se kreće u smeru sve veće vrednosti - ona praktično *planinari*. Kada se dođe do pozicije iznad koje ne postoji sused sa većim vrhom, ona se zaustavlja sa radom (dostignut je lokalni maksimum) - planinarenje ne gleda dalje od neposrednih suseda. [111] U tom slučaju, postavlja se nova slučajna početna vrednost i algoritam na osnovu nje pokušava da dođe do krajnjeg rešenja. Važno je napomenuti da se ne generiše stablo pretrage, tako da se u trenutnom koraku čuvaju informacije vezane samo za taj korak. [112, 113]

Planinarenje spada u pohlepne (*greedy*) algoritme koji lokalno traže najbolje rešenje nadajući se da će ono biti i globalno najbolje. Uz razuman broj koraka, ovakvi algoritmi mogu doći do rešenja, koje često ne mora biti optimalno. [114] Na primer, može se posmatrati problem trgovačkog putnika u okviru koga je potrebno naći najkraći put kojim trgovački putnik treba da obiđe niz gradova, ali tako da kroz svaki prođe samo jednom. Pohlepni algoritam će funkcionisati tako što će u svakom koraku posetiti samo najbliži grad onome u kome se trenutno nalazi. Ovaj problem ima veliku složenost i nalaženje idealnog rešenja može da bude komplikovano. Pohlepni algoritam će prilično lako i sa umerenim brojem koraka doći do rešenja, ali je veoma verovatno da ono neće biti optimalno.

Kao čest primer za objašnjavanje upotrebe algoritma planinarenje koristi se primer n -kraljica (n -queens). Osnova je šahovska tabla dimenzija $n \times n$, na kojoj je potrebno rasporediti kraljice, ali tako da se međusobno nijedna ne napada. Jednostavan primer sa četiri kraljice prikazan je na grafikonu 36.

Grafikon 36: Jednostavan primer n -kraljica.



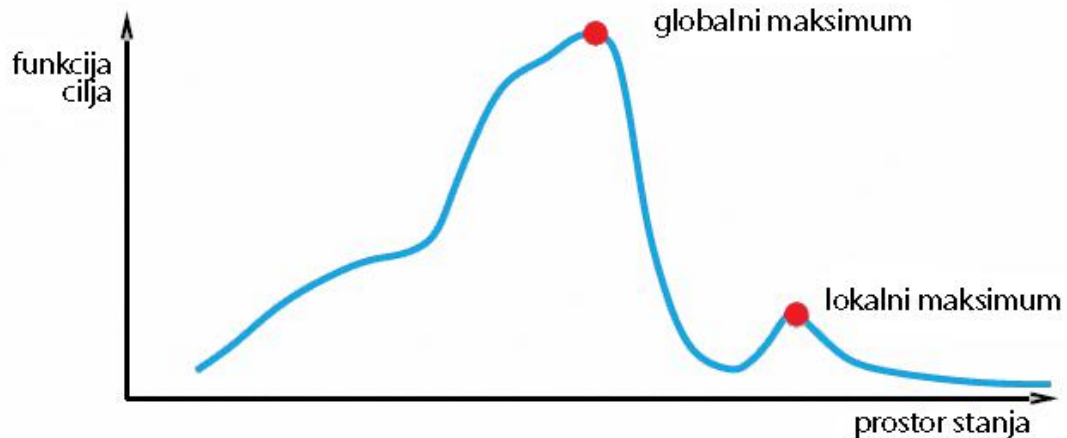
Prilagođeno prema: Indiana University, Artificial Intelligence Course [115]

Kao što je već navedeno, prvi korak je da se kraljice nasumično rasporede. Zbog slučajnog rasporeda, velika je verovatnoća da će se neke međusobno napadati, budući da dele kolonu, red ili dijagonalu. Zatim se nasumično izabere jedna od kraljica koje su napadnute. U okviru reda u kome se nalazi, pomera se na neku poziciju na kojoj je niko neće napadati. Ukoliko takva pozicija ne postoji, onda se bira pozicija koju napada najmanji broj kraljica. Ukoliko su nakon pomeranja neke kraljice i dalje napadnute, onda se jedna od njih nasumično bira i pomera na novu lokaciju. Cilj je da se broj konflikata na tabli minimizuje.

Osnovna strategija ovog algoritma se pokazuje izuzetno uspešnom. U stanju je da reši problem n -kraljica za slučaj sa milion kraljica za manje od jednog minuta na standardom računaru. [116] Treba imati u vidu da je ovaj test napravljen pre više od dvadeset godina i da bi sa današnjim računarima rezultat bio još bolji.

Pretraživanje planinarenjem može naići i na probleme. Konkretno, u toku pretrage, algoritam može doći do lokalnog maksimuma, koji predstavlja vrh koji je viši u odnosu na svoju okolinu, ali je niži u odnosu na globalni maksimum (grafikon 37) - to je obično tačka od koje više neće moći da se ostvari napredak.

Grafikon 37: Problemi sa lokalnim maksimumima u planinarenju.



Problem predstavlja i to što planinarenje, kao algoritam lokalne pretrage, nema mogućnost da detektuje kada rešenje ne postoji - u ovakvoj situaciji, algoritam će praktično nastaviti da u nedogled pravi lokalne izmene [117]. U određenim slučajevima je bolje i započeti algoritam od novog slučajnog početka nakon što posle određenog vremena rešenje ne bude pronađeno i algoritam dospe u slepu ulicu (npr. kada se dostigne lokalni maksimum). Pokazuje se da je to mnogo efikasnije nego da se dozvoli da algoritam beskonačno pravi izmene. Jedna moguća realizacija ovog problema je da se ograniči broj lokalnih izmena koje se mogu napraviti. [118]

Mana algoritma je što se ne može garantovati optimalnost pronađenog rešenja. Kao rešenje tog problema, mogao bi se uvesti mehanizam koji bi pamtio sva prethodna stanja i obezbeđivao da se ne istražuju stanja koja su već istraživana. U tom slučaju, pretraga bi u jednom trenutku mogla da istraži ceo prostor problema, ali imajući u vidu kombinatornu prirodu algoritma, postavlja se pitanje upotrebljivosti takvog pristupa. Takođe, ne postoji ni način da se u okviru algoritma precizno podesi odnos između brzine i kvaliteta rada. Često se dešava da algoritam i ne pronađe rešenje, ili da ono ne bude optimalno.

3.4.4. A* pretraživanje

A* pretraživanje spada u najpoznatiji oblik pretraživanja prvo najbolji (*best-first search*) sa kojim u osnovi deli istu proceduru rada. Spada u najpoznatije algoritme za probleme nalaženje puta jer je fleksibilan i može se koristiti u velikom broju situacija, uključujući i rad sa ogromnim prostorima stanja. Prvi put je predstavljen 1968. godine, a predstavljao je nadogradnju Dijkstrinog algoritma iz 1959. godine [119]. Dobre performanse su postignute korišćenjem heurističkih funkcija.

Algoritam A* koristi funkciju $f(n)$ kako bi odredio redosled u kom će se obilaziti graf. Funkcija $f(n)$ predstavlja sumu druge dve funkcije [120]:

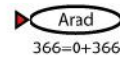
$$f(n) = g(n) + h(n).$$

Funkcija $g(n)$ predstavlja funkciju stvarnog troška, od početnog do trenutnog čvora, dok funkcija $h(n)$ predstavlja *heurističku* funkciju - funkciju koja procenjuje preostalo rastojanje do cilja. Važno je da $h(n)$ bude *prihvatljiva* heuristika koja nikada neće preceniti trošak stizanja do cilja. [121]

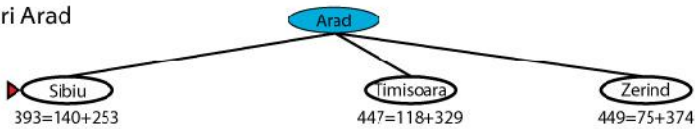
Grafikon 38 grafički prikazuje način rada algoritma A* na primeru mape rumunskih gradova. Konkretno, potrebno je doći od Arada do Bukurešta. Kao heuristička funkcija $h(n)$ koristi se pravolinijska udaljenost do odredišta koja predstavlja prihvatljivu heuristiku jer predstavlja najkraće rastojanje do cilja i ne može biti precenjena. U okviru grafa, svaka grana mora imati definisanu težinu, odnosno u ovom slučaju kilometražu.

Grafikon 38: Prikaz rada algoritma A* na primeru mape rumunskih gradova.

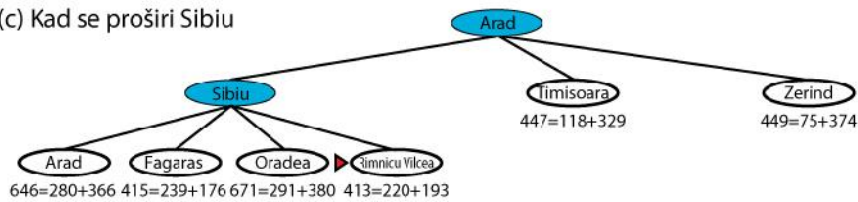
(a) Početno stanje



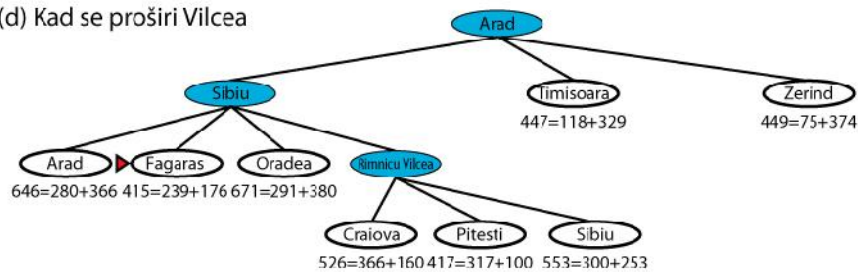
(b) Kad se proširi Arad



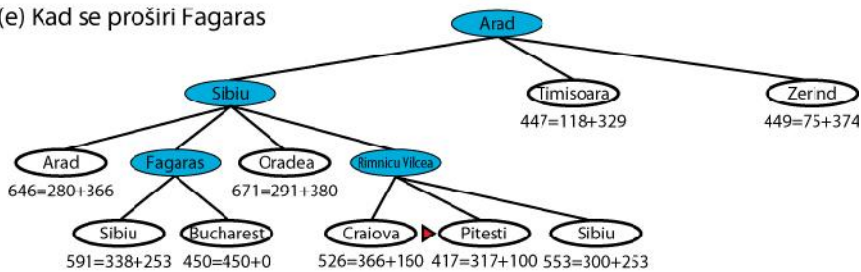
(c) Kad se proširi Sibiu



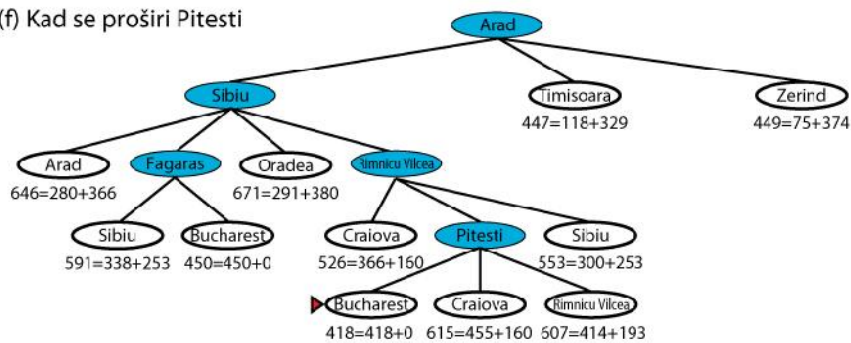
(d) Kad se proširi Vilcea



(e) Kad se proširi Fagaras



(f) Kad se proširi Pitesti



Prilagođeno prema: <http://aima.cs.berkeley.edu/>

Algoritam A* održava listu čvorova koji se razmatraju za otvaranje (otvorena lista) i listu čvorova koji su već otvarani (zatvorena lista). U svakom koraku, najbolji čvor iz otvorene liste se pomera u zatvorenu listu, potom se proširuje i njegovi sledbenici se dodaju u otvorenu listu. Ta procedura se nastavlja sve dok se ne dođe do rešenja. Uz konzistentnu heurističku funkciju, čvor koji je proširen i premešten u zatvorenu listu, nikada ne bi trebao da bude ponovo vraćen u otvorenu listu. [122] Sam algoritam ima sledeću strukturu [69]:

1. Formirati listu parcijalnih putanja. Inicijalno lista sadrži samo jednu putanju nulte dužine koja koja sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste putanja koja dostiže ciljni čvor.
 - 2.1. Ako je prva putanja dostigla ciljni čvor, ne raditi ništa.
 - 2.2. Ako prva putanja nije dostigla ciljni čvor, uraditi sledeće:
 - 2.2.1. Ukloniti prvu putanju iz liste.
 - 2.2.2. Za svaki sledbenik poslednjeg čvora na uklonjenoj putanji formirati po jednu novu putanju produžujući sledbenikom uklonjenu putanju.
 - 2.2.3. Za svaku od novodobijenih putanja izračunati ukupnu (kumulativnu) cenu koštanja c kao zbir cena koštanja operatora na toj putanji; za poslednji čvor na putanji izračunati heurističku funkciju h . Funkciju procene f za svaku od novih putanja izračunati kao zbir heurističke funkcije h i cene koštanja putanje c ($f = h + c$).
 - 2.2.4. Dodati nove putanje u listu parcijalnih putanja.
 - 2.2.5. Sortirati listu putanja po rastućim vrednostima funkcije procene f .
 - 2.2.6. Ako dve ili više putanja iz liste imaju isti poslednji čvor, ukloniti iz liste sve takve putanje osim jedne koja ima najmanju cenu koštanja (princip dinamičkog programiranja).
3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

A* je kompletan algoritam koji će pronaći rešenje problema ukoliko ono postoji. Ukoliko ne uspe da pronađe rešenje, tada se može garantovati da ono i ne postoji. A* će naći rešenje uz najmanji mogući trošak, a to će veoma zavisiti od definisane heurističke funkcije i njene procene. [123] U slučaju kada je heuristička funkcija nekonzistentna, algoritam će raditi veoma loše jer će se određeni čvorovi koji su već otvarani otvarati ponovo veliki broj puta. To u najgorem slučaju može rezultovati složenošću od $O(2^N)$, gde je N broj različitih čvorova koji su otvarani. [124]

3.5. Predlog rešenja softverskog sistema

U okviru ovog poglavlja definisana je osnova novog softverskog simulatora. Zahvaljujući korišćenju dve klase algoritama - mašinskog učenja i pretraživanja - pojavila se prirodna podela sistema na dve logičke celine. Svaka od ovih celina biće zadužena za jednu klasu algoritama koji su prethodno obrađeni, a u okviru sebe će sadržati pod-segmente (module) zadužene za pojedinačne algoritme.

Deo zadužen za algoritme mašinskog učenja će biti podeljen na četiri modula: stabla odlučivanja, klasterovanje, Naive Bayes i perceptroni. S obzirom da su ovi algoritmi dosta različiti po načinu rada, u njihovom izgledu i korišćenju će postojati određene razlike, uz obavezno vođenje računa o funkcionalnoj usaglašenosti sa ostatkom simulatora.

Deo sistema koji je namenjen algoritmima pretraživanja neće biti podeljen na module, već će činiti jedinstvenu celinu u okviru koje će postojati mogućnost rada sa grafovima i stablima, nakon čega će se birati algoritam koji će biti primenjen. Ovakvo integrisano rešenje biće moguće izvesti zbog strukturne sličnosti algoritama pretraživanja.

Ceo softver će imati isti način organizacije i rada, zahvaljujući kome će, čak i kod algoritama koji strukturno ne dele zajedničku osnovu, postojati utisak ujednačenosti i poznatog okruženja pri korišćenju. Svaki segment će imati određene obavezne celine koje su prilagođene potrebama trenutnog algoritma:

- *Unos početnih podataka (ulaz algoritma)* - kod stabala odlučivanja i Naive Bayes-a početni podaci će biti predstavljeni nakon tekstualnog unosa iz fajla, kod

klasterovanja i perceptrona će se definisati direktno u simulatoru, dok će se kod algoritama pretraživanja unositi kroz odgovarajući dijalog crtanjem grafa ili stabla (ili unosom iz tekstualnog fajla).

- *Grafički izlaz algoritma* će biti najvažniji deo simulatora i omogućavaće da se rad algoritma vizuelno prikaže i potom prati na ekranu. Zbog različitosti algoritama, u okviru ovog dela postojaće najveće razlike u prikazivanju između pojedinačnih delova simulatora.
- *Tekstualni izlaz algoritma* pružaće podršku grafičkom prikazu pomoću dodatnih objašnjenja o tome šta se dešava u svakom koraku algoritma.
- *Komande simulatora* omogućavaće korisniku upravljanje softverom. Obavezni deo za svaki segment biće linija sa glavnim alatima na vrhu ekrana, kao i pomoćne komande vezane za određene algoritme: uvećavanje i umanjivanje grafova i stabala, ažuriranje ulaznih setova podataka, brisanje tekstualnih poruka itd.

4. REALIZACIJA NOVOG SOFTVERSKOG SIMULATORA

Sistem je podeljen na dve logičke celine. Svaka od njih je zadužena za simulaciju jedne vrste algoritama - algoritama mašinskog učenja i algoritama pretraživanja. Nakon pokretanja sistema, prikazuje se početni prozor i meniji. Iz menija *Simulator*, korisnik može izabrati koju simulaciju želi da pokrene. Nakon što napravi izbor, učitava se odgovarajuća radna površina, kao i potrebne linije sa alatkama.

Za simulaciju svakog algoritma, prilagođena je posebna radna površina, u skladu sa njegovim potrebama. Pored obaveznih delova, kao što su grafički prikaz simulacije i okvir za ispis poruka, svaka radna površina može imati i neke neobavezne delove kao što su, na primer, polja za unos dodatnih podataka.

Kako je mogućnost postepenog kretanja kroz algoritam izuzetno važna, postoje i obavezna dugmad pomoću kojih se korisniku omogućavaju te funkcionalnosti:

- *napred, nazad* – ova dugmad omogućavaju pokretanje algoritma za jedan korak unapred ili unazad;
- *na početak* – pomoću ove naredbe se izvršenje algoritma vraća na početak;
- *na kraj* – pomoću ove naredbe se prelazi na poslednji korak algoritma.

U daljem tekstu, funkcionalnost svakog modula biće bliže predstavljena uz detaljnije navođenje njegovih mogućnosti i karakteristika.

4.1. Prikaz funkcionalnosti sistema

Sistem je podeljen na module, a svaki pojedinačni modul je zadužen za simulaciju jednog algoritma, osim u slučaju algoritama pretraživanja. Zbog njihove međusobne sličnosti, ovi algoritmi se nalaze u okviru jednog modula u okviru koga se bira koji će se algoritam koristiti.

U okviru modula za stabla odlučivanja, korisnik ima mogućnost da iz tekstualnog dokumenta učitava primere za obučavanje. Dokument u kome se nalaze test primeri je jednostavan *Tab delimiter file*, u okviru koga svaki red predstavlja novi zapis, a pojedinačni elementi u okviru toga zapisa se razdvajaju *tab*-om. Ovaj pristup je pogodan i zbog toga što je moguće ulazne primere izvesti direktno iz nekog programa za obradu podataka kao što je *Microsoft Excel*. Na osnovu tih primera, u simulatoru će biti generisana tabela sa ulaznim podacima kako bi korisnik mogao da ih lakše pregleda. Na osnovu tih podataka, algoritam će obučiti stablo. Takođe, postoji i mogućnost pravljenja novog test primera unosom željenih atributa i njihovih vrednosti u tabelu. Korisnik unete podatke može izvesti i u tekstualni fajl kako bi ih sačuvao ili dalje distribuirao.

U cilju lakšeg razumevanja, samo stablo se iscrtava u posebnom okviru. Atributi su predstavljeni čvorovima stabla, dok su vrednosti atributa predstavljene granama. Desnim klikom na čvorove, prikazuje se informaciona dobit svih atributa. Takođe, tokom iscrtavanja svake grane stabla, u okviru za poruke se ispisuju vrednosti entropije svakog čvora. Omogućeno je i uveličavanje i umanjivanje prikaza stabla, kao i njegovo pomeranje, kako bi korisnik mogao da pronađe način pregledanja koji mu najviše odgovara.

Za demonstraciju rada algoritma za klasterovanje koristi se određeni broj tačaka koje je potrebno grupisati. Korisnik ima mogućnost da utiče na broj klastera, broj tačaka, kao i stepen raspršenosti tačaka. Kako bi bio olakšan unos tih vrednosti upotrebljeni su klizači pomoću koji se jednostavno mogu uneti željene vrednosti. Takođe, klizači su se pokazali i kao pogodan način da se korisnički unos ograniči samo na određeni skup vrednosti – npr. na brojeve od 2 do 10.

U zavisnosti od unetih parametara za generisanje tačaka i klastera, iscrtavaju se tačke koje služe za demonstraciju algoritma. Pre pokretanja algoritma, svaka tačka se predstavlja krstićem crne boje, dok se početni centroidi predstavljaju punim krugovima crvene boje. Nakon pokretanja algoritma, svaka tačka će biti označena krstićem u boji - boja zavisi od toga kom klasteru pripada. Sami centroidi će biti predstavljeni punim krugovima, a njihova boja će takođe zavisiti od toga kom klasteru pripadaju.

Ukoliko se označi opcija *prikaz istorije*, biće prikazane i sve prethodne pozicije centroida. U ovom slučaju, grafički prikaz se blago menja. Punim krugom će biti označene sve prethodne pozicije, dok se trenutno stanje obeležava kvadratom. Na ovaj način se može videti na koji način su se centriodi menjali dok nisu došli do krajnje pozicije.

Postepenim prolaskom kroz algoritam, u okviru za poruke će biti ispisane informacije o trenutnom koraku – njegov broj, kao i broj tačaka koje su promenile klaster.

U okviru modula za algoritam Naive Bayes, korisnik ima mogućnost da iz tekstualnog dokumenta učitava primere za obučavanje koji će se koristiti. Na osnovu tih primera, generiše se tabela sa ulaznim podacima kako bi mogao lakše da se izvrši pregled unetih vrednosti. Na osnovu tih vrednosti, biće izvršeno obučavanje algoritma. Takođe, kao i u slučaju stabla odlučivanja, postoji mogućnost direktnog unosa potrebnih vrednosti, bez učitavanja iz datoteke.

Korisniku se prikazuje i dodatna tabela pomoću koje se bira test primer. Vrednosti za svaki atribut će se nalaziti u padajućoj listi u kojoj će se isključivo nalaziti vrednosti iz ulaznog seta. Ukoliko se promeni neka od ovih vrednosti, potrebno je kliknuti na dugme *Osvežavanje test primera*, kako bi novoizabrani atributi bili postavljeni kao podrazumevani.

Izlaz algoritma će se prikazivati u vidu stabla u okviru koga će svaki čvor predstavljati zasebnu uslovnu verovatnoću. Koren tog stabla sadrži krajnje rešenje, tj. određenu verovatnoću test primera (obeleženu sa V_{NB}). U svakom koraku, u ulaznoj tabeli označavaju se redovi koji se koriste za obračun. Detaljne informacije o svakom koraku ispisuju se u posebnom tekstualnom okviru.

U okviru modula za neuronske mreže simulira se rad perceptrona kao osnovne gradivne jedinice neuronskih mreža. Vršiti se izbor početnih vrednosti – u svakom paru po dve ulazne i jedna izlazna. Skup brojeva koje ove vrednosti mogu uzeti čine 0 i 1. Na ovaj način se prikazuje rad perceptrona sa Bulovim operacijama. Takođe, moguće je uneti i *stepen učenja* (α), kao i *prag* (θ). U svakom koraku se označava red čije se vrednosti u tom trenutku koriste kao ulaz i izlaz perceptrona.

U svakoj iteraciji algoritma grafički se prikazuju ulazne vrednosti perceptrona, odgovarajuće težine, stepen učenja, prag i izlazna vrednost. Kao i kod ostalih modula, i u ovom slučaju postoji okvir za tekst u okviru koga se ispisuju detaljne informacije o svakom koraku.

Podsistem za rad sa algoritmima pretraživanja ima funkcionalno ujednačeniji interfejs od dela koji je zadužen za mašinsko učenje. Razlog je u tome što su izabrani algoritmi mašinskog učenja dosta raznovrsni i u velikoj meri između sebe različiti, za razliku od algoritama pretraživanja. Ovi algoritmi dele sličnu osnovu rada - svaki od njih za početak rada zahteva graf (ili stablo kao podvrstu grafa), nad njim potom izvršava određene operacije, iscrtava stablo pretraživanja i prikazuje rezultat. Zbog toga je interfejs za sva četiri algoritma pretraživanja praktično identičan. Glavne razlike se nalaze u samom radu izabranih algoritama.

Rad sa algoritmima pretraživanja počinje crtanjem grafa. Korisnik ima opcije da unosi pojedinačne čvorove, menja njihove pozicije, definiše veze između (grafovi mogu biti povezani, nepovezani, ciklični itd.), težine pojedinih grana, vrednosti čvorova, kao i početnu i krajnju tačku. Nakon toga, pokreće se algoritam i korak po korak prolazi kroz graf tražeći rešenje. Na dnu ekrana se iscrtava stablo pretraživanja koje prikazuje posećene čvorove, počevši od početnog. Na taj način, korisnik ima mogućnost da prati na koji način algoritam radi. U okviru za poruke se, kao i u podsistemu za mašinsko učenje, prikazuju detaljne informacije o aktivnostima koje se dešavaju kako bi se imao bolji uvid u pozadinu rada algoritma.

U toku rada, može se manipulirati grafičkim prikazom grafa i stabla pretrage - mogu im se menjati pozicija i uvećanje - na taj način, prikaz se u svakom trenutku maksimalno može prilagoditi krajnjem korisniku.

4.2. Tehnologije korišćene za izradu sistema

Novi softverski sistem predstavljen u ovom radu zasnovan je na Java platformi. Java je programski jezik otvorenog koda i predstavlja globalni standard za razvoj aplikacija nezavisnih od platforme na kojoj se izvršavaju. Jedna od najvažnijih karakteristika jezika je arhitekturna neutralnost. [125]

Java platforma obuhvata čitav niz grafičkih biblioteka koje omogućavaju dodatne funkcionalnosti i koje se koriste u različite svrhe. Za potrebe razvoja ovog simulatora, vođeno je računa da se izabere pravi sistem koji će moći da omogući najveći skup mogućnosti i pouzdanost u radu. U praksi su se kao najkorisnija rešenja pokazali:

- *AWT* koji predstavlja osnovu Swing-a, ali mu nedostaju naprednije mogućnosti. Predstavlja stabilno rešenje koje se dokazalo u praksi.
- *Swing* je u početku bio relativno spor i problematičan, ali su sa novijim verzijama prevaziđeni problemi. Izuzetno se često koristi za razvoj aplikacija jer poseduje veliki broj komponenti za bogat korisnički interfejs.
- *SWT* je nastao kao odgovor kompanije IBM na Swing. Iako u odnosu na njega nudi neke dodatne komponente, nije se pokazao kao stabilan i popularan za rad.
- *SwingX* je baziran na Swing-u. Ima određen broj kvalitetnih komponenti, ali je zvanično još uvek u razvoju.
- *JavaFX* predstavlja obećavajući standard za razvoj desktop i veb aplikacija. Java FX je sledeći korak u evoluciji Java programa sa bogatim korisničkim interfejsima i naslednik Swing-a koji trenutno ima veći broj komponenti u odnosu na JavaFX.

Za potrebe simulatora, biblioteka AWT je izabrana za iscrtavanje komponenti na najnižem nivou, počevši od pravljenja najosnovnijih oblika. Swing komponente su korišćene za standardne elemente korisničkog interfejsa (paneli, dugmad, meniji, polja za unos podataka itd.). Kao dodatni alat za rad sa grafovskim strukturama, upotrebljena je biblioteka JUNG [129]. Izborom ovakve kombinacije softvera omogućeno je razvijanje grafički bogate

aplikacije koja se koristi kao desktop rešenje, ali se isto tako veoma brzo može pretvoriti u aplet i ugraditi u veb stranu, kako bi je bilo moguće lakše distribuirati pomoću interneta.

U narednim odeljcima ove biblioteke su opisane, a njihove tehničke karakteristike su detaljnije predstavljene.

4.2.1. Java AWT

Java AWT (*Abstract Window Toolkit*) je biblioteka programskog jezika Java zadužena za osnovno programiranje grafičkog korisničkog interfejsa i predstavlja vezu ka matičnom korisničkom interfejsu sistema na kome se izvršava.

Korisnički interfejs je deo programa koji obezbeđuje interakciju sa korisnikom programa. Na najnižem nivou, operativni sistem prenosi informacije od miša i tastature do programa kao ulaz, i obezbeđuje piksele za izlaz programa. AWT je osmišljen tako da programeri ne moraju da brinu o praćenju miša ili čitanju sa tastature ili da brinu o ispisu na ekranu. AWT obezbeđuje dobro dizajnirani objektno orijentisani interfejs ka servisima i resursima niskog nivoa. [126]

Najvažnije karakteristike AWT-a su [127]:

- veliki broj komponenti za realizaciju korisničkog interfejsa,
- robustan model za rukovanje događajima,
- alati za rad sa slikama i grafičkim elementima (uključujući klase za rad sa oblicima, bojama, fontovima),
- menadžeri rasporeda za fleksibilan rad sa prozorima
- klase za prenos podataka - za korišćenje *clipboard*-a na matičnoj platformi.

AWT komponente u velikoj meri zavise od platforme na kojoj se program izvršava. Na primer, pravljenjem neke komponente korisničkog interfejsa pomoću AWT-a, biće direktno pozvana odgovarajuća sistemska rutina koja će obezbediti zahtevanu komponentu. To znači da će program na Windows-u imati izgled kao izvorne Windows aplikacije ili da će imati izgled Macintosh aplikacije kada se pokrene na ovom operativnom sistemu. Naravno, ovakav

pristup može biti problematičan ukoliko je potrebno obezbediti da interfejs izgleda identično na svim platformama.

4.2.2. Java Swing

Biblioteka Swing obuhvata veliki set komponenti za izradu grafičkih korisničkih interfejsa i dodavanje interaktivnosti Java aplikacijama. Swing obuhvata veliki broj komponenti koje bi mogle da se koriste u jednoj modernoj aplikaciji: tabele, liste, stabla, dugmad, natpise... Pored toga, obuhvata i odličnu podršku za rad sa tekstom, kao i integrisanu podršku za internacionalizaciju i dostupnost. Swing poseduje i mnoštvo opcija za podešavanje *izgleda i osećaja* (*look and feel*), a postoji i mogućnost pravljenja sopstvenog. Pored svega pobrojanog, postoji i podrška za *povuci i pusti* tehnike (*drag and drop*), rukovanje događajima, prilagođeno iscrtavanje i rukovanje prozorima. [128]

Swing je razvijen kako bi omogućio sofisticiraniji skup komponenti za razvijanje korisničkih interfejsa. Sve ove komponente nisu implementirane u kodu koji je specifičan za određenu platformu, već su kompletno napisane u Javi zbog čega su platformski nezavisne.

Od J2SE 1.2 Swing je u velikoj meri potisnuo AWT. Pored obezbeđivanja bogatog skupa ulazno-izlaznih komponenti, Swing prikazuje i sopstvene dodatke koji se iscrtavaju koristeći Java 2D pomoću koga se pozivaju systemske rutine niskog nivoa u lokalnom grafičkom podsistemu. Na taj način je izbegnuto oslanjanje na module visokog nivoa za korisnički interfejs u okviru operativnog sistema. Swing obezbeđuje mogućnost izbora *izgleda i osećaja* koji će se koristiti - može se upotrebiti ili onaj koji je platformski vezan, ili međuplatformski *izgled i osećaj* koji izgleda isto na svim sistemima za prikaz prozora.

Naravno, i pored toga što je Swing stekao izuzetno veliku popularnost, ipak ne predstavlja zamenu za AWT, već njegovu nadogradnju, s obzirom da je napravljen na njegovim bibliotekama.

Pre pojavljivanja Java 6 Update 12, kombinovanje Swing komponenti sa AWT dodacima je često dovodilo do nepredviđenih efekata, kada bi se AWT dodaci pojavljivali preko onih definisanih u Swing-u, bez obzira na navedeni *z-raspored*. Ovi problemi su nastajali zbog

toga što su dva navedena sistema u suštini veoma različita, i pored toga što Swing pozajmljuje neke kontejnere višeg nivoa od AWT-a.

Swing predstavlja izuzetno modularnu platformu, a omogućava i "priključivanje" dodatnih, namenski pravljenih implementacija za potrebe specifičnih interfejsa. Podrazumevane komponente se mogu modifikovati nasleđivanjem klasa, čime se mogu dodati određene nove funkcionalnosti ili izmeniti postojeće.

4.2.3. Biblioteka JUNG

S obzirom da je u određenim modulima potrebno iscrtavati stabla koja služe kao osnova za grafičko reprezentovanje rada algoritama, potrebno je obezbediti način da se to obavi. Jedna mogućnost je ručno pravljenje mehanizama koji bi omogućio podršku za iscrtavanje, a druga da se upotrebi postojeća biblioteka koja može da pruži potrebne funkcionalnosti i čijim korišćenjem može da se uštedi vreme potrebno za razvoj.

Kao odlično rešenje može se upotrebiti biblioteka JUNG [129] - *Java Universal Network/Graph Framework*. Ova biblioteka obezbeđuje proširiv jezik za modelovanje, analizu i vizuelizaciju podataka koji mogu biti predstavljeni u obliku mreže ili grafa. Dizajnirana je tako da podrži varijacije prikaza određenih entiteta i njihovih veza, kao što su usmereni i neusmereni grafovi ili grafovi sa paralelnim granama. Zahvaljujući ovoj biblioteci, programeri ne moraju da brinu o grafičkoj implementaciji samih grafova ili stabala jer posao prikazivanja ovih elemenata preuzima JUNG.

Najvažnije karakteristike biblioteke JUNG su [130]:

- Podrška za razne reprezentacije entiteta i njihovih veza, uključujući usmerene i neusmerene grafove, multimodalne grafove, multi-grafove i hiper-grafove.
- Mehanizmi za anotaciju grafova, entiteta i veza pomoću metapodataka.
- Postojanje implementacije određenog broja algoritama iz teorije grafova i analize podataka.

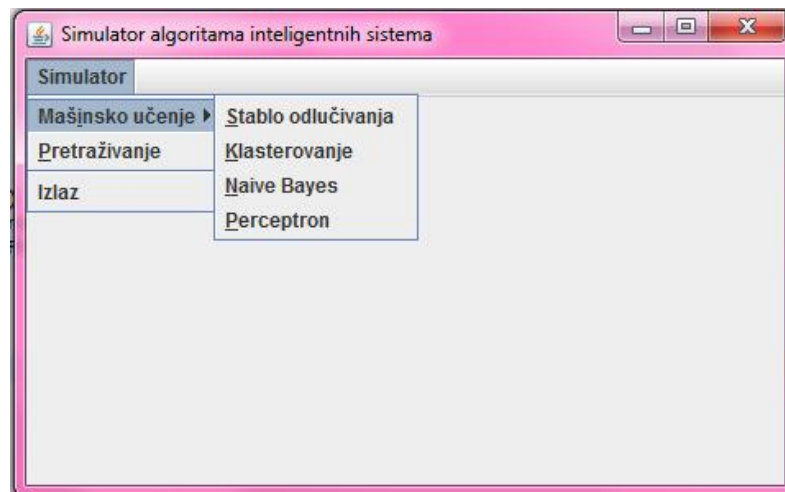
- Alati za vizuelizaciju koji olakšavaju razvoj mehanizama za interaktivno pretraživanje i prikazivanje grafovskih podataka.
- Mehanizmi za filtriranje pomoću kojih se može doći do određenog podskupa grafovske mreže koja se prikazuje, kako bi se na njemu, na primer, primenio određeni algoritam.

4.3. Opis funkcionalnosti sistema

Ova sekcija predstavlja detaljan opis funkcionalnosti sistema. Kroz prikaze ekrana i tekstualna obrazloženja, može se steći uvid u način rada realizovanog simulatora.

Nakon pokretanja simulatora, prikazuje se prazan početni ekran uz mogućnost izbora željene simulacije u meniju (grafikon 39). Tek nakon što se izabere algoritam koji će biti simuliran, odgovarajući modul će biti pokrenut. Grafikon 40 prikazuje izgled ekrana simulatora nakon što je izabran rad sa stablom odlučivanja - prikazan je meni koji se koristi u ovom modulu, kao i paneli za unos i ispis podataka u toku rada algoritma.

Grafikon 39: Početni ekran aplikacije bez ijednog učitanoog modula.



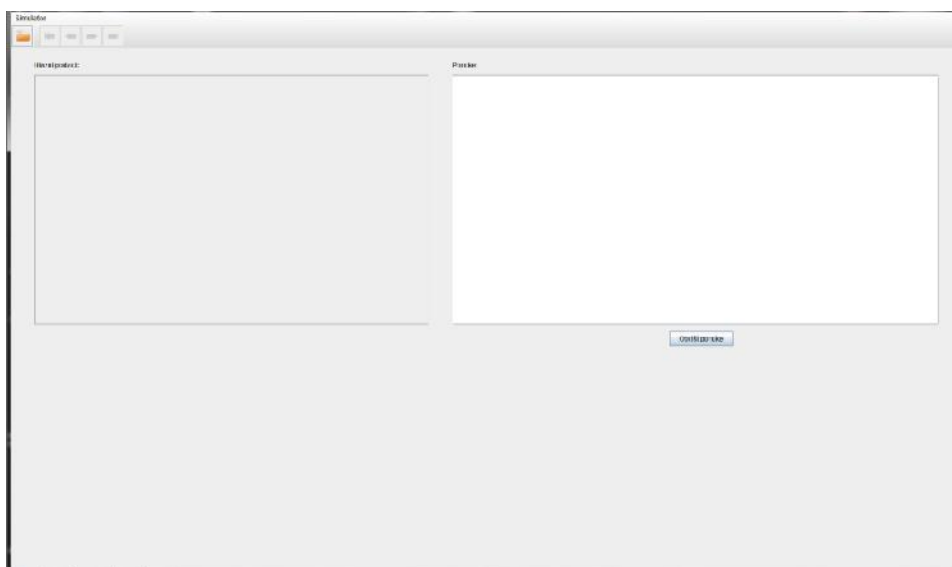
U daljem tekstu, funkcionalnost svakog modula biće detaljnije predstavljena.

4.3.1. Opis funkcionalnosti stabla odlučivanja

Modul stablo odlučivanja sastoji se iz tri celine - prikaza tabele sa nazivima atributa i ulaznim podacima, prikaza poruka o radu algoritma i grafičkog prikaza stabla odlučivanja. Izgled modula stablo odlučivanja prikazan je na grafikonu 40.

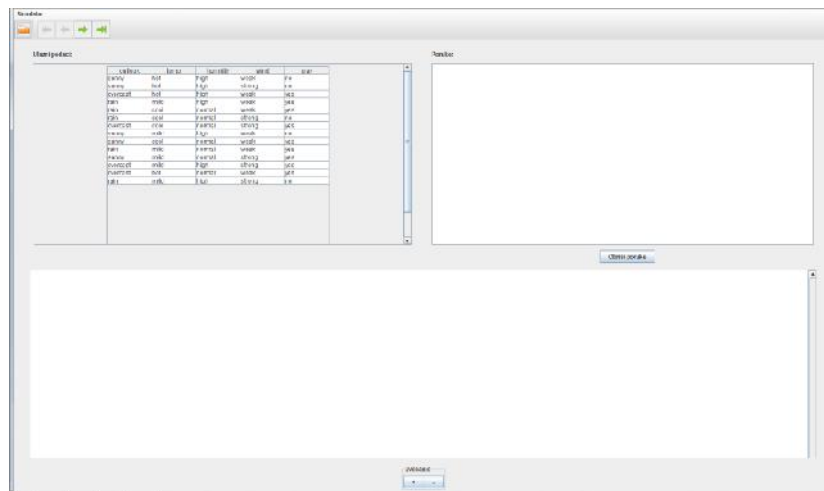
Korisnik ima mogućnost otvaranja dokumenta sa ulaznim skupom podataka - u prvom redu se nalaze nazivi atributa razdvojeni tabom. U svakom sledećem redu se nalaze vrednosti ulaznih atributa, takođe razdvojene tabom.

Grafikon 40: Izgled modula stablo odlučivanja.



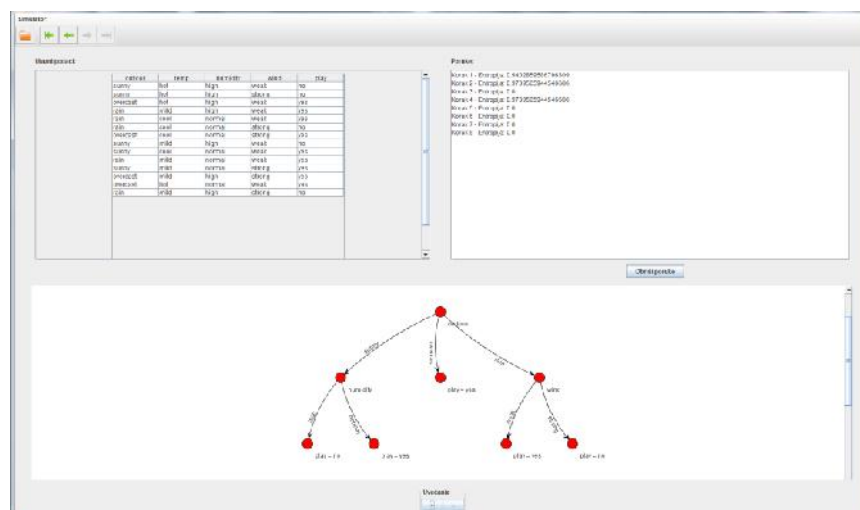
Nakon učitavanja podataka, vrednosti se smeštaju u odgovarajuću tabelu, a opcije za kretanje kroz algoritam postaju omogućene (grafikon 41).

Grafikon 41: Izgled modula nakon unosa podataka.



Nakon pokretanja algoritma, u posebnom okviru se iscrtava sâmo stablo odlučivanja (grafikon 42). Poruke o svakom koraku ispisuju se u panelu sa desne strane - prikazuju se vrednosti entropije svakog čvora.

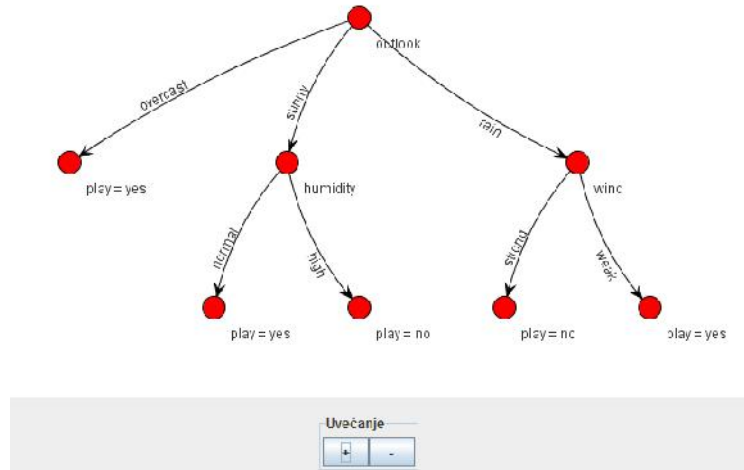
Grafikon 42: Prikaz generisanog stabla odlučivanja.



Ukoliko se detaljnije pogleda način na koji se iscrtava stablo (grafikon 43), primećuje se da se pored svakog čvora prikazuje naziv tog atributa, dok se pored grana koje idu od tog do sledećeg čvora prikazuju vrednosti tog atributa. Listovi stabla sadrže odluku koju donosi stablo odlučivanja. Ukoliko se desnim tasterom klikne na neki čvor, u kontekstnom meniju

će biti prikazane vrednosti informacione dobiti svakog od čvorova dece. Moguće je vršiti uvećavanje ili umanjivanje stabla, kao i njegovo pomeranje u okviru panela.

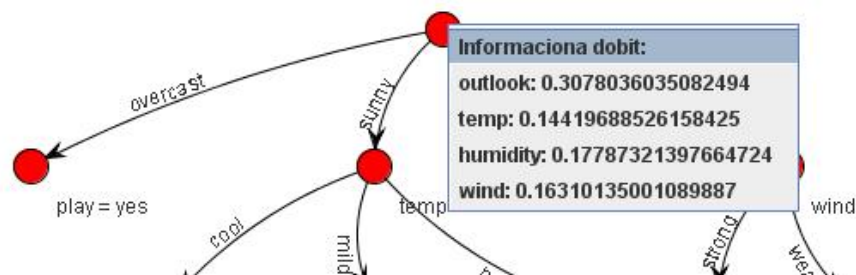
Grafikon 43: Izgled stabla odlučivanja.



Bitno je naglasiti da algoritam *ID3* ne mora uključiti sve ulazne atribute u krajnje stablo odlučivanja. Ukoliko donese takvu odluku, može izostaviti neke čvorove ukoliko proceni da nisu važni u donošenju odluke.

Ukoliko se desnim tasterom miša klikne na neki od čvorova, u kontekstnom meniju će biti prikazane vrednosti informacione dobiti, na osnovu kojih se može videti na koji način je algoritam izvršio podelu ulaznog skupa (grafikon 44).

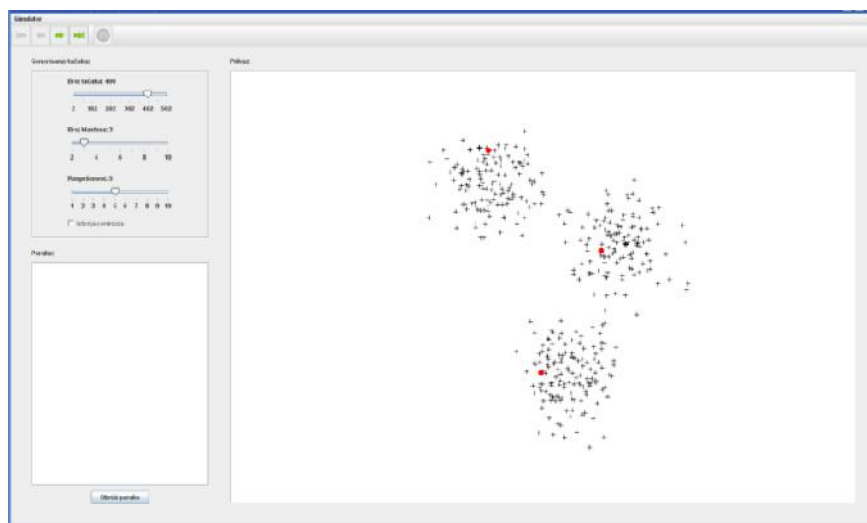
Grafikon 44: Prikaz informacionih dobiti u kontekstnom meniju.



4.3.2. Opis funkcionalnosti algoritma klasterovanje

Modul za klasterovanje takođe ima tri dela kao i stablo odlučivanja. U prvom delu se rukuje prikazom tačaka, kao i istorijom pomeranja centroida, u drugom se vrši ispis poruka, a u trećem se iscrtavaju same tačke, kao i klasteri sa odgovarajućim centroidima. Na grafikonu 45 prikazan je izgled modula klasterovanje.

Grafikon 45: Izgled modula klasterovanje.



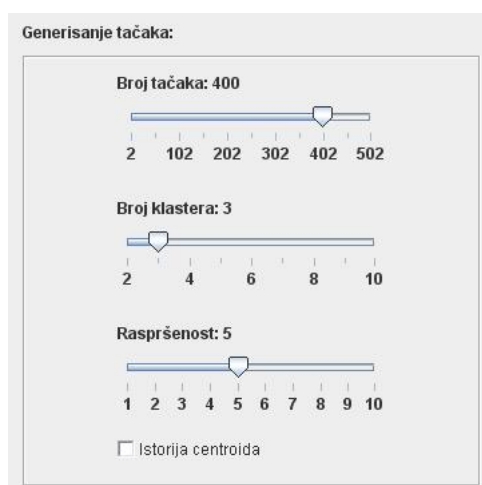
Pre početka rada algoritma, potrebno je podesiti određene parametre (grafikon 46):

- inicijalni broj tačaka;
- broj klastera;
- raspršenost tačaka.

Da bi se ograničio korisnički unos, koriste se klizači koji su pogodni da se lakše definiše koje vrednosti korisnik može uneti.

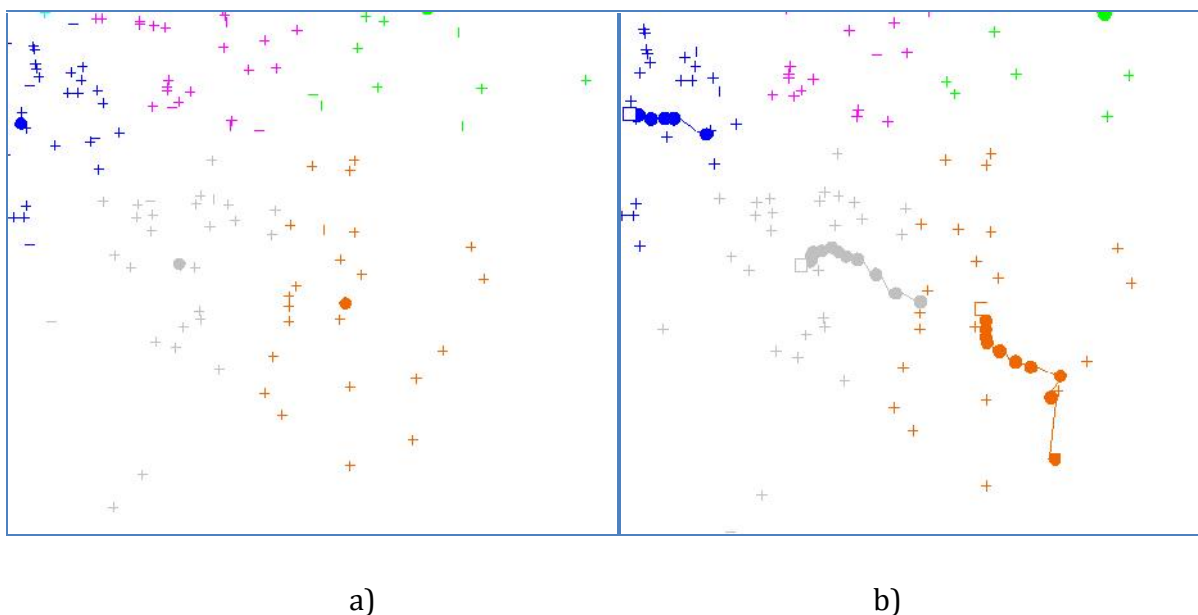
Važno je napomenuti da broj klastera ne sme biti veći od broja tačaka. Ukoliko se pokuša takav odabir parametara, biće prikazana upozoravajuća poruka.

Grafikon 46: Podešavanje početnih vrednosti.



Postoji i opcija za aktiviranje prikaza istorije. Ukoliko se ova opcija aktivira, pored trenutnog centroida, biće prikazane i sve prethodne pozicije povezane linijama kako bi se mogla videti putanja koju je centroid „prešao“. Na grafikonu 47 a) prikazan je deo prikaza zajedno sa nekoliko centroida. Ukoliko se aktivira prikaz istorije, dobiće se prikaz kao na grafikonu 47 b). U ovom slučaju, jasno se vidi kako su se menjale pozicije centroida.

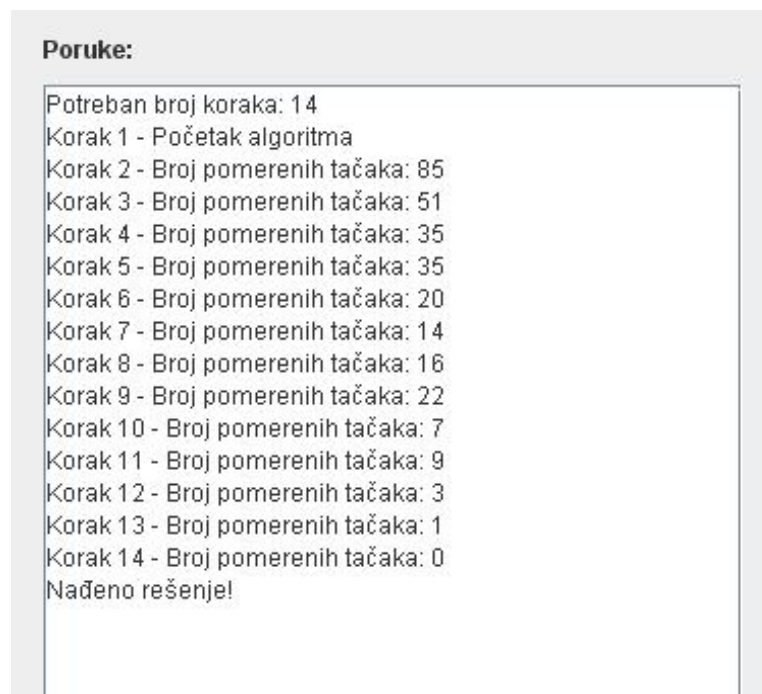
Grafikon 47: Prikaz istorije centroida.



Bitno je napomenuti da se za nasumično generisanje tačaka koristi funkcija *random*, ali sa fiksim *seed*-om. Na taj način je obezbeđeno da se svaki put sa istim odabranim vrednostima za broj tačaka, klastera, kao i raspršenost, dobiju potpuno identične pozicije tačaka.

Na grafikonu 48 prikazan je izgled okvira za poruke. Na samom početku rada biće prikazano u koliko će koraka algoritam stići do rešenja. Potom se ispisuje broj pomerenih tačaka u svakom prolazu.

Grafikon 48: Prikaz tekstualnog izlaza algoritma.



Da bi se izmenile početne vrednosti, potrebno je zaustaviti rad algoritma pritiskom na dugme *Zaustavljanje simulacije* u liniji alata (grafikon 49).

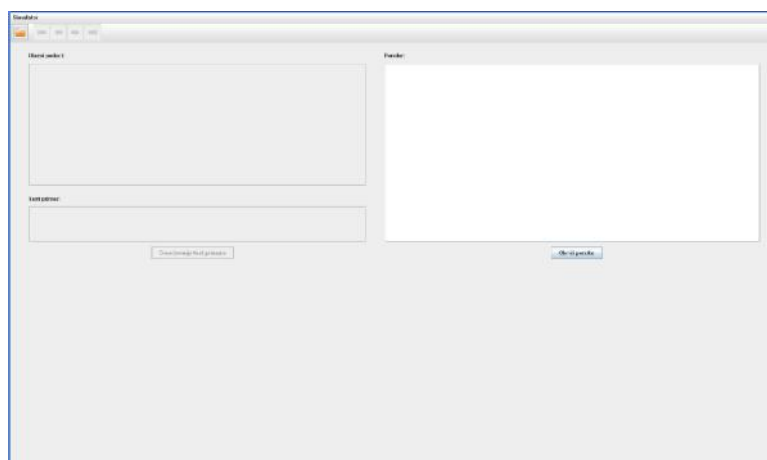
Grafikon 49: Linija alata za modul klasterovanje.



4.3.3. Opis funkcionalnosti algoritma Naive Bayes

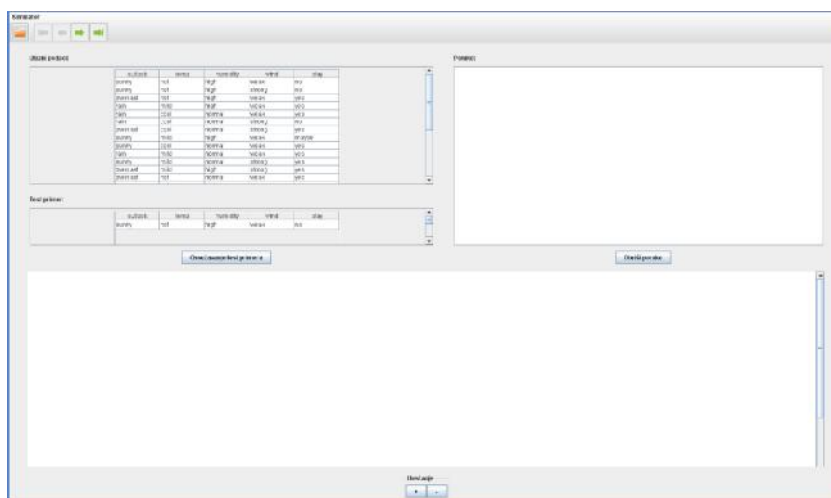
Modul Naive Bayes poseduje nekoliko osnovnih delova: panele sa ulaznim i test podacima i okvire za poruke i grafički izlaz algoritma. Na grafikonu 50 prikazan je početni izgled ovog modula.

Grafikon 50: Izgled modula Naive Bayes.



Da bi se počelo sa radom, potrebno je učitati tekstualni dokument ili uneti podatke. Ukoliko se odabere unos preko tekstualnog dokumenta, podaci će biti učitani u tabelu sa ulaznim podacima (grafikon 51).

Grafikon 51: Izgled modula Naive Bayes nakon učitavanja početnih podataka.



U tabelu sa test podacima biće smeštene postojeće vrednosti atributa u okviru padajućih lista. Iz svake od lista, korisnik može izabrati željene attribute na osnovu kojih će algoritam obračunati verovatnoće. Ukoliko se izmeni neki od atributa, potrebno je kliknuti na dugme *Osvežavanje test primera* da bi se stari test primer uklonio iz prikaza, a novi primer učitao (grafikon 52).

Grafikon 52: Izbor željenog test primera.

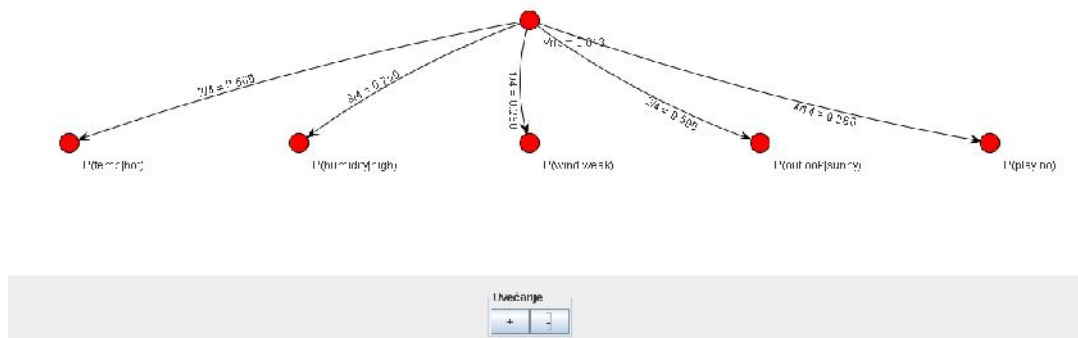
Test primer:

outlook	temp	humidity	wind	play
sunny	hot	high	weak	no
			weak	
			strong	

Osvežavanje test primera

Nakon odabira željenog test primera, može se pokrenuti algoritam. sa svakim korakom algoritma, u okviru za grafički prikaz se iscrtavaju čvorovi koji predstavljaju uslovne verovatnoće. Na grafikonu 53 je dat krajnji rezultat izvršavanja jednog primera.

Grafikon 53: Izgled stabla sa uslovnim verovatnoćama.



Svaki list prikazuje koju uslovnu verovatnoću predstavlja, dok se na grani koja ga povezuje sa korenom ispisuje vrednost te uslovne verovatnoće iskazana kao broj odgovarajućih primera podeljen ukupnim brojem primera. U korenu stabla nalazi se odredišna verovatnoća, a označena je sa V_{NB} .

U toku rada algoritma, u tabeli sa ulaznim podacima se u svakom koraku označavaju redovi koji su uzeti u obračun (grafikon 54).

Grafikon 54: Označavanje redova u tabeli sa ulaznim podacima.

Ulazni podaci:

outlook	temp	humidity	wind	play
sunny	hot	high	weak	no
sunny	hot	high	strong	no
overcast	hot	high	weak	yes
rain	mild	high	weak	yes
rain	cool	normal	weak	yes
rain	cool	normal	strong	no
overcast	cool	normal	strong	yes
sunny	mild	high	weak	maybe
sunny	cool	normal	weak	yes
rain	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes

Svaki korak se detaljno objašnjava u polju za ispis poruka (grafikon 55).

Grafikon 55: Prikaz izlaza algoritma Naive Bayes.

Poruke:

Korak 1
 $P(\text{play}|\text{no}) = 4 / 14 = 0.286$

Korak 2
 $P(\text{outlook}|\text{sunny}) = 2 / 4 = 0.500$

Korak 3
 $P(\text{temp}|\text{hot}) = 2 / 4 = 0.500$

Korak 4
 $P(\text{humidity}|\text{high}) = 3 / 4 = 0.750$

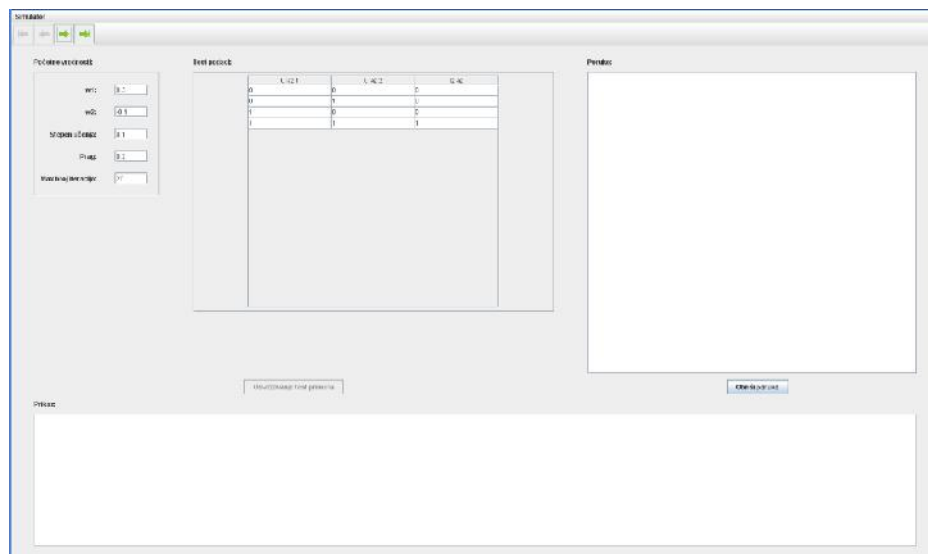
Korak 5
 $P(\text{wind}|\text{weak}) = 1 / 4 = 0.250$

Korak 6
Pronađeno rešenje
 $V_{nb} = 0.013$

4.3.4. Opis funkcionalnosti perceptrona

Modul perceptron sastoji se od okvira za podešavanje ulaznih parametara, tabele sa ulaznim podacima, okvira za ispis tekstualnih poruka i okvira za grafičku predstavu perceptrona. Izgled modula prikazan je na grafikonu 56.

Grafikon 56: Izgled modula perceptron.



U okviru za određivanje početnih vrednosti, moguće je podesiti inicijalne vrednosti težine ($w1$ i $w2$), stepen učenja algoritma, prag perceptrona i maksimalni broj iteracija rada programa (grafikon 57).

Grafikon 57: Podešavanje ulaznih parametara algoritma.

Početne vrednosti:

w1:	0.3
w2:	-0.1
Stepen učenja:	0.1
Prag:	0.2
Max broj iteracija:	20

U tabeli sa ulaznim podacima se kao ulaz koristi binarna funkcija za potrebe simuliranja osnovnih Bulovih operacija. Postoje dva ulaza i izlaz. Vrednosti izlaza se mogu menjati pomoću padajuće liste – dozvoljene vrednosti su 0 i 1. Izgled tabele prikazan je na grafikonu 58.

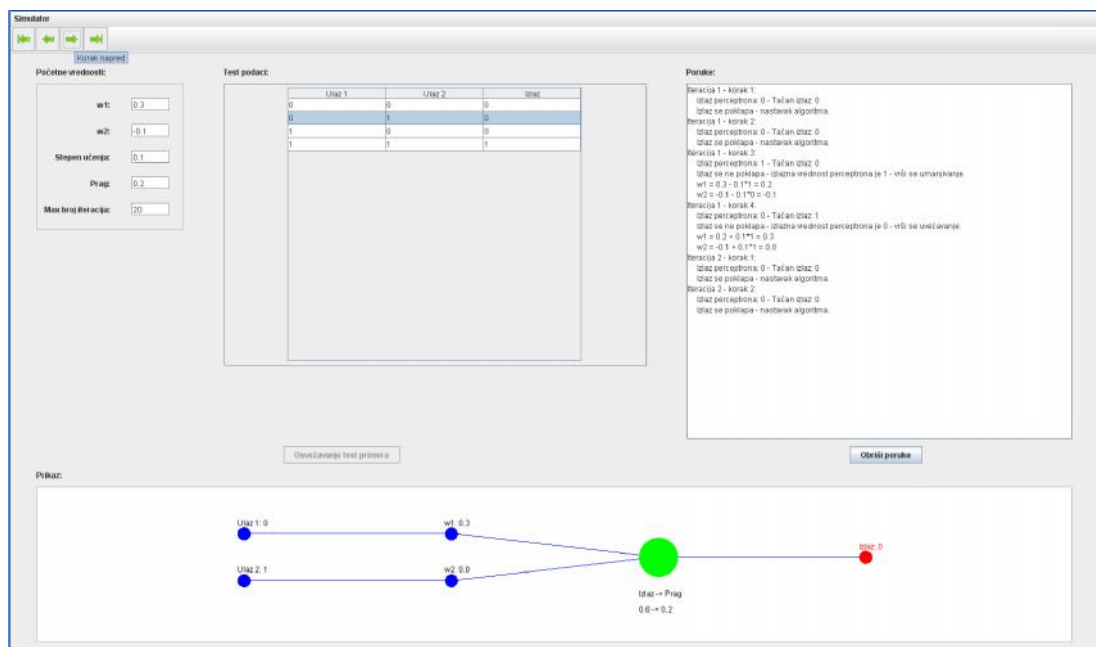
Grafikon 58: Izbor početnog skupa vrednosti.

Test podaci:

Ulaz 1	Ulaz 2	Izlaz
0	0	0
0	1	0
1	0	0
1	1	1
		0
		1

Nakon podešavanja ovih vrednosti, moguće je pokrenuti algoritam. Grafikon 59 prikazuje izgled modula nakon pokretanja.

Grafikon 59: Izgled modula nakon pokretanja algoritma.



Algoritam uzima red po red iz tabele sa ulaznim podacima i te vrednosti prosleđuje na obradu perceptronu. Red sa kojim se trenutno radi je označen u odnosu na druge redove (grafikon 60).

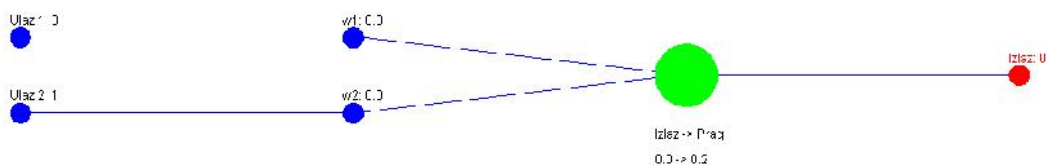
Grafikon 60: Označavanje reda koji se trenutno koristi.

Test podaci:

Ulaz 1	Ulaz 2	Izlaz
0	0	0
0	1	0
1	0	0
1	1	1

Perceptron prihvata vrednosti iz ulazne tabele i obračunava izlaz, a grafička prezentacija njegovog rada prikazana je na grafikonu 61. Prikazuju se ulazne vrednosti, težine, suma ulaznih vrednosti i težina, prag i izlaz. Radi bolje organizacije i lakšeg razumevanja, određeni delovi grafikona su prikazani različitim bojama.

Grafikon 61: Grafički prikaz perceptrona.



U svakom koraku, ispisuje se i tekstualni izlaz programa u okviru za poruke (grafikon 62). Prikazuju se trenutna iteracija, kao i korak u okviru te iteracije. Prikazuje se i da li je došlo do poklapanja izlaza perceptrona i željenog izlaza. Ukoliko nije došlo do poklapanja, prikazuje se na koji način se obračunavaju promene težine i koje su njihove vrednosti.

Grafikon 62: Tekstualni izlaz algoritma.

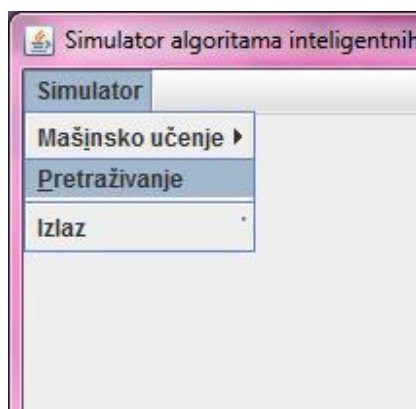
```
Poruke:
Iteracija 1 - korak 1:
  Izlaz perceptrona: 0 - Tačan izlaz: 0
  Izlaz se poklapa - nastavak algoritma.
Iteracija 1 - korak 2:
  Izlaz perceptrona: 0 - Tačan izlaz: 0
  Izlaz se poklapa - nastavak algoritma.
Iteracija 1 - korak 3:
  Izlaz perceptrona: 1 - Tačan izlaz: 0
  Izlaz se ne poklapa - izlazna vrednost perceptrona je 1 - vrši se umanjivanje.
  w1 = 0.3 - 0.1*1 = 0.2
  w2 = -0.1 - 0.1*0 = -0.1
Iteracija 1 - korak 4:
  Izlaz perceptrona: 0 - Tačan izlaz: 1
  Izlaz se ne poklapa - izlazna vrednost perceptrona je 0 - vrši se uvećavanje.
  w1 = 0.2 + 0.1*1 = 0.3
  w2 = -0.1 + 0.1*1 = 0.0
Iteracija 2 - korak 1:
  Izlaz perceptrona: 0 - Tačan izlaz: 0
  Izlaz se poklapa - nastavak algoritma.
Iteracija 2 - korak 2:
  Izlaz perceptrona: 0 - Tačan izlaz: 0
  Izlaz se poklapa - nastavak algoritma.
```

4.3.5. Opis funkcionalnosti modula za algoritme pretraživanja

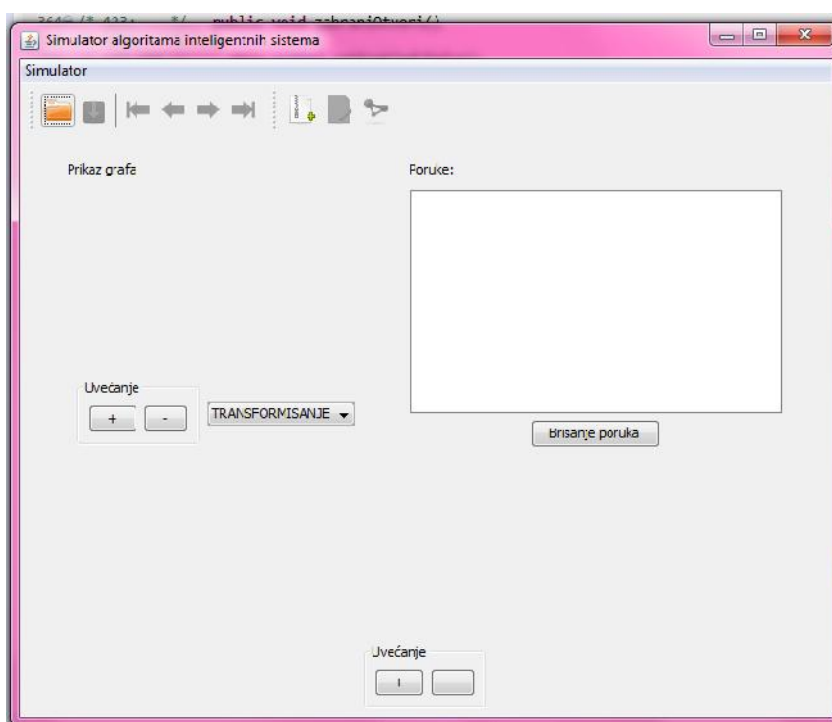
U okviru modula za algoritme pretraživanja, realizovana su četiri algoritma pretraživanja, koja su i pomenuta u prethodnom delu teksta. Za razliku od algoritama mašinskog učenja koji su međusobno veoma različiti zbog čega je opis rada svakoga od njih morao da bude pojedinačno naveden, algoritmi pretraživanja u osnovi dele istu strukturu. Jedino što je specifično za svaki od njih jeste način rada. Zbog toga, u ovoj sekciji neće biti objašnjavan način rada svakog od njih pojedinačno, već će biti predstavljen ceo modul, uz navođenje određenih specifičnosti svakog od algoritama.

Nakon što korisnik na početnom ekranu izabere modul za simulaciju algoritama pretraživanja (grafikon 63), otvoriće se početni ekran koji sadrži obavezne elemente korisničkog interfejsa ovog modula (grafikon 64).

Grafikon 63: Izbor algoritama pretraživanja.



Grafikon 64: Početni izgled modula za algoritme pretraživanja.



Na početku rada, korisnik ima mogućnost učitavanja postojećeg grafa direktno iz tekstualnog fajla, a graf koji je napravljen ili izmenjen u simulatoru može se i sačuvati u posebnom dokumentu - prve dve opcije na grafikonu 65. Strelice orijentisane u levo i desno predstavljaju opcije za kretanje kroz algoritam korak po korak, ili za prebacivanje na početak ili kraj algoritma. Poslednje tri opcije služe za pravljenje novog grafa ukoliko nije učitano

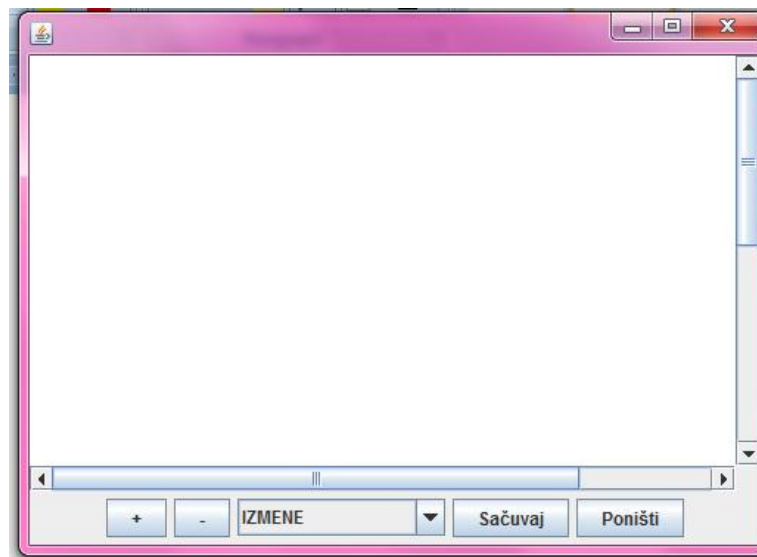
nijedan postojeći, za menjanje već napravljenog grafa, kao i za izbor algoritma pretraživanja koji će biti upotrebljen.

Ukoliko se klikne na dugme za pravljenje novog grafa, biće otvoren prazan dijalog u okviru koga se može crtati novi graf (grafikon 66). Ukoliko je već učitana postojeća graf iz fajla, ili je napravljen novi graf u simulatoru, klikom na dugme za izmene se može pristupiti menjanju grafa (grafikon 67). U ovom dijalogu, korisnik ima mogućnost dodavanja čvorova, veza između čvorova, menjanja njihovih atributa, kao i definisanja početne i krajnje tačke algoritma.

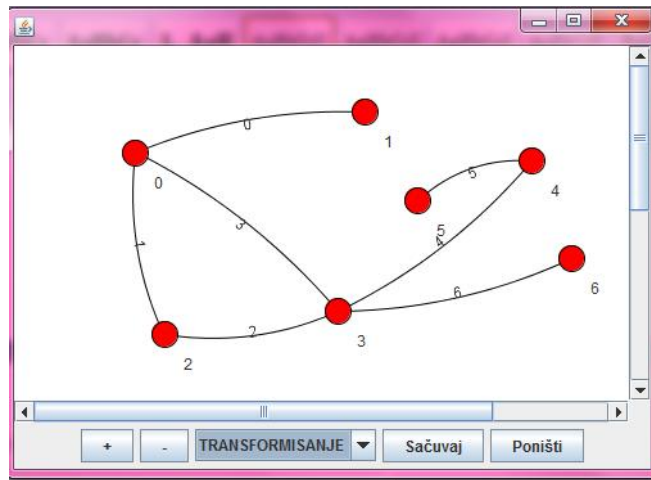
Grafikon 65: Linija sa alatima u modulu Pretraživanje.



Grafikon 66: Okvir za pravljenje novog grafa.

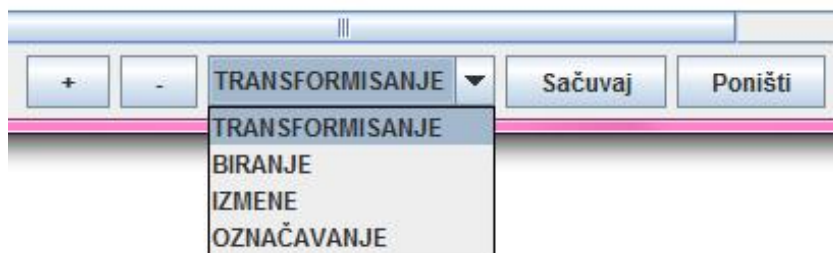


Grafikon 67: Izgled okvira za izmenu postojećeg grafa.



Na dnu ekrana za pravljenje i pregled grafova (grafikoni 66 i 67) nalazi se padajuća lista sa opcijama za njegove izmene, kao što je prikazano na grafikonu 68.

Grafikon 68: Padajuća lista sa opcijama za izmene grafa.



Korisnik može izabrati sledeće opcije:

- "Transformisanje" omogućava pomeranje grafa u okviru dijaloga, kao i promenu njegovog uvećanja.
- "Biranje" omogućava izbor svakog pojedinačnog čvora ili veze, pri čemu se mogu menjati vrednosti koje su im dodeljene, kao i njihove fizičke pozicije u grafu (grafikon 69).
- "Izmene" omogućava dodavanje novih čvorova u grafu, kao i novih veza.
- "Označavanje" pruža mogućnost dodavanja grafičkih i tekstualnih oznaka u određene delove grafa, ukoliko je neki njegov deo potrebno dodatno istaći.


```

graph TD
    A((A)) --> B((B))
    A((A)) --> C((C))
    B((B)) --> D((D))
    C((C)) --> D((D))
    D((D)) --> C((C))
    D((D)) --> E((E))
    E((E)) --> B((B))

```

- 103 -

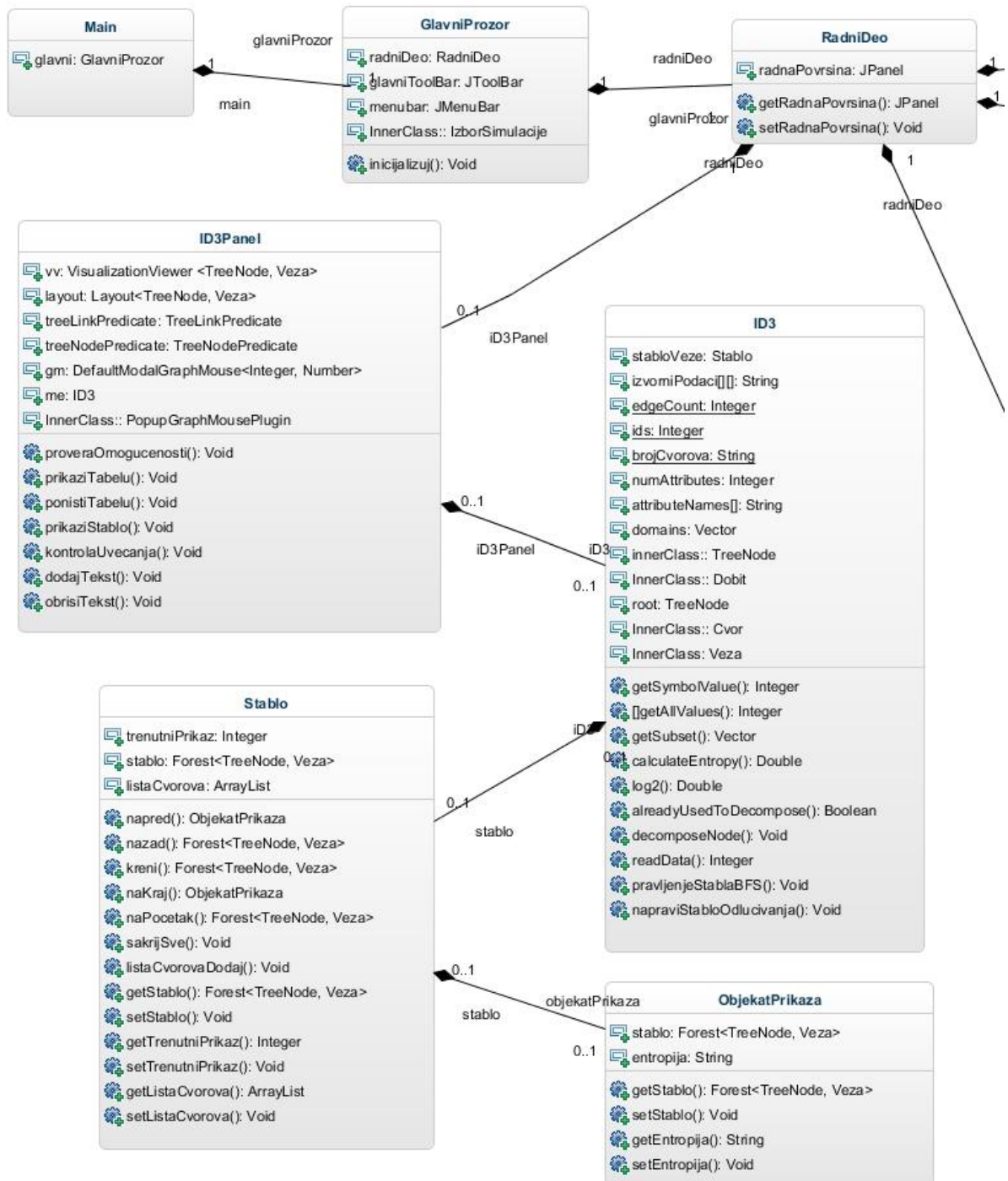
4.4. Implementacija sistema

Programski kod sistema se sastoji iz dve celine (algoritmi mašinskog učenja i algoritmi pretraživanja) i raspoređen je u nekoliko paketa. Glavni program od koga počinje izvršavanje nazvan je *Main* nalazi se u paketu *main*. Klase koje rukuju prikazom, radnom površinom i rasporedom komponenti, nalaze se u paketu *gui*. Klase za rad sa algoritmima su grupisane u odgovarajuće pakete nazvane po samim algoritmima.

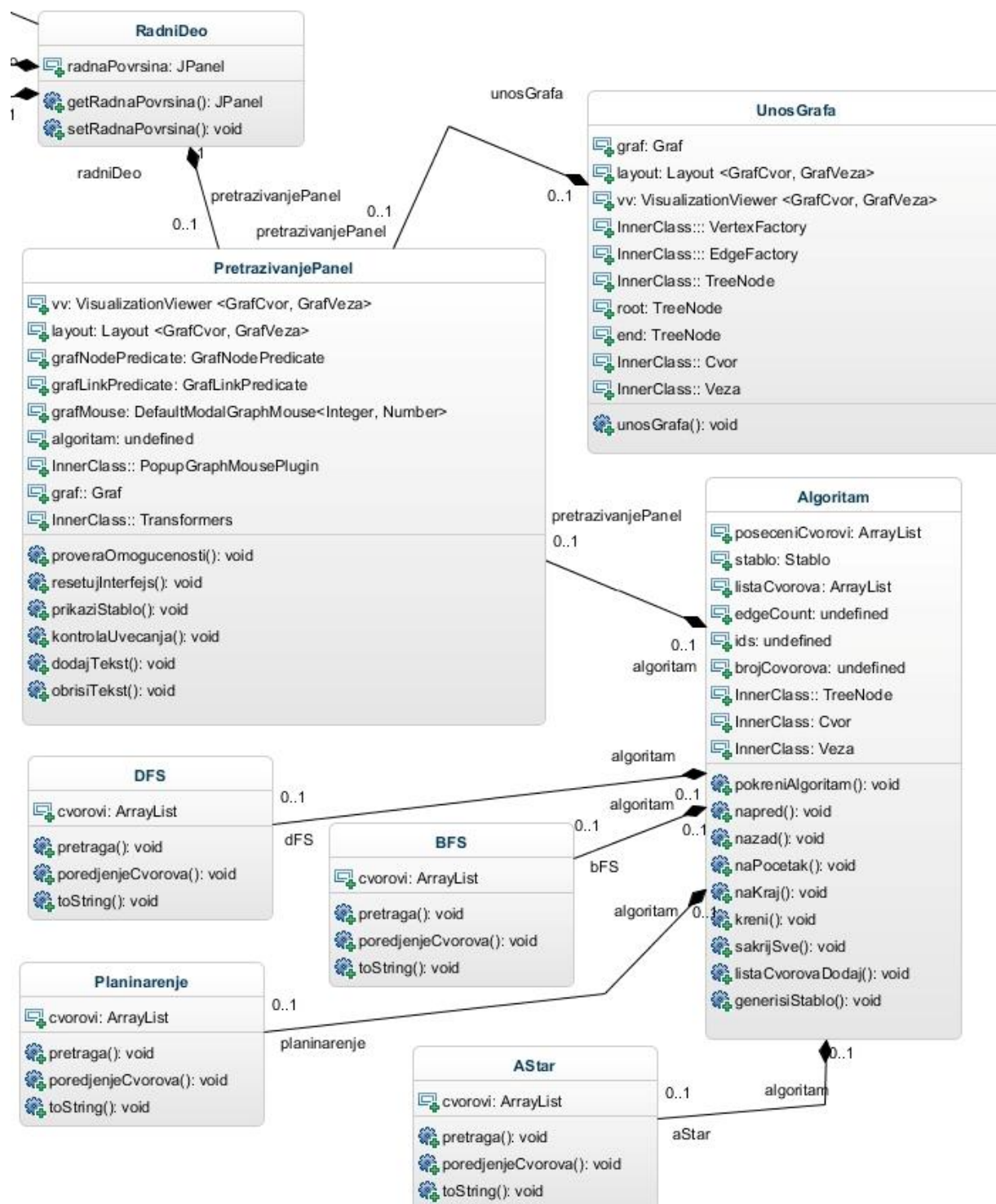
Grafikon 72 prikazuje dijagram klasa softverskog simulatora. Prikazana je osnovna struktura, a navedene su i unutrašnje klase, čije dekompozicije zbog obimnosti nisu prikazane u dijagramima. Polja i metodi iz paketa Java Swing vezani za prikaz korisničkog interfejsa (JPanel, JTextArea, JScrollPane, JLabel, JButton itd.) u cilju povećanja preglednosti nisu prikazani u dijagramu klasa, s obzirom da su vezani samo za tehničku implementaciju interfejsa.

U narednim odeljcima su pojašnjeni svi moduli i neki od problema koji su rešeni prilikom realizacije, a tiču se same strukture simulatora, najvažnijih klasa, kao i rada algoritama.

Grafikon 72: Dijagram klasa softverskog simulatora.



Grafikon 72: Dijagram klasa softverskog simulatora - nastavak.



4.4.1. Glavni prozor programa

Glavni prozor programa napravljen je korišćenjem klase *GlavniProzor.java* koja sadrži programsku logiku za prikaz samog okvira prozora, menija, linije alata, ali i izmenljive radne površine - postavljanje ovih komponenti zavisi od trenutno aktivnog modula. Ova klasa vrši i inicijalno postavljanje grafičkog korisničkog interfejsa pomoću funkcije *inicijalizuj()*. Prikaz dela programskog koda prikazan je u listingu 1.

Listing 1: Najvažnija polja klase *GlavniProzor.java*.

```
public class GlavniProzor extends JFrame {  
    private RadniDeo radniDeo;  
    private JPanel toolbar;  
    private JToolBar glavniToolBar;  
    private JMenuBar menubar;  
    private JMenu simulator;  
    private JMenu simulacija;  
    private JMenu izborSimulacije;  
    ...  
    public GlavniProzor() {  
        inicijalizuj();  
    }  
    ...  
}
```

Polje *radniDeo* ima važnu ulogu budući da je zaduženo je za čuvanje reference ka modulu koji će trenutno biti prikazan u prozoru programa. Ovo polje pripada klasi *RadniDeo.java*. Ova klasa čuva trenutni modul kao običan *JPanel* objekat i na taj način rukuje sa njim. Postoji nekoliko panel klasa, po jedna za svaki od algoritama mašinskog učenja (*id3Panel*, *kmPanel*, *nbPanel* i *perceptronPanel*) i jedna objedinjujuća za algoritme pretraživanja koji dele sličnu osnovu rada (*pretrazivanjePanel*).

Nakon što se pokrene simulator, prikazuje se prazan početni ekran uz mogućnost izbora željene simulacije u meniju. Tek nakon što se izabere algoritam koji će biti simuliran, odgovarajući modul će biti pokrenut i sačuvan kao *JPanel* objekat. Grafikon 40 prikazuje izgled ekrana simulatora nakon što je izabran rad sa stablom odlučivanja - prikazan je meni koji se koristi u ovom modulu, kao i paneli za unos i ispis podataka u toku rada algoritma.

U narednim odeljcima, prikazani su detalji implementacije za svaki od modula simulatora.

4.4.2. Implementacija stabla odlučivanja

S obzirom da algoritam *ID3* u svojoj osnovi ima strukturu podataka stablo, veoma je važno predstaviti čvor tog stabla. Listing 2 prikazuje kako izgleda početak klase *TreeNode*.

Listing 2: Osnovna struktura klase *TreeNode*.

```
public class TreeNode {
    public double entropy;
    public String entropijaTekst;
    public ArrayList<String> data;
    public int decompositionAttribute;
    public int decompositionValue;
    public TreeNode[] children;
    public TreeNode parent;
    public int id;
    private boolean vidljivo = false;
    ArrayList<Dobit> dobiti;

    public String poruka;

    public TreeNode() {
        data = new ArrayList<String>();
        id = ids;
        ids++;
        entropijaTekst = "";
        dobiti = new ArrayList<Dobit>();
    }
}
```

Neka od najvažnijih polja ove klase su:

- *entropy* čuva vrednost entropije određenog čvora. Ukoliko se radi o listu stabla, entropija će imati vrednost 0.
- *decompositionAttribute* predstavlja atribut po kom je izvršena podela stabla. Ovaj atribut bira algoritam stabla odlučivanja kao najbolji za podelu.
- *decompositionValue* predstavlja vrednost izabranog atributa po kojoj je izvršena podela ulaznog skupa podataka.
- *children* čuva reference ka svim čvorovima deci trenutnog čvora. ukoliko se radi o listu stabla, ovo polje neće imati vrednost.

- *vidljivo* je atribut koji se koristi kod grafičke prezentacije stabla. S obzirom na to da se zbog postupnog prolaska kroz algoritam svi čvorovi ne prikazuju odjednom, pomoću ovog atributa će se označiti koji će biti vidljivi, a koji ne. Na ovaj način će biblioteci JUNG biti saopšteno kako treba da prikaže stablo.
- *dobiti* čuva informacione dobiti atributa. Ove vrednosti će biti prikazane u kontekstnom meniju koji se pojavljuje kada se desnim tasterom miša klikne na neki čvor.

Klasa *ID3* zadužena je za rukovanje samim stablom odlučivanja. U okviru nje se prolazi kroz proces podele ulaznog skupa podataka i pravljenja odredišnog stabla. Kao polje, u okviru nje se čuva *root* stabla. Sa ovim elementom povezani su svi sledeći čvorovi.

Da bi se napravljeno stablo moglo prikazati, mora se prilagoditi za korišćenje JUNG-a. Za tu svrhu je napravljena klasa *Stablo* čija je skraćena struktura prikazana u listingu 3.

Listing 3: Osnovna struktura klase *Stablo*.

```
public class Stablo {
    private int trenutniPrikaz;
    Forest<TreeNode, Veza> stablo;
    private ArrayList<ID3.TreeNode> listaCvorova;

    public Stablo() {
        trenutniPrikaz = -1;
        listaCvorova = new ArrayList<ID3.TreeNode>();
        stablo = new DelegateTree<TreeNode, Veza>();
    }

    public ObjekatPrikaza napred() {
        ...
        return op;
    }

    public Forest<TreeNode, Veza> nazad() {
        ...
        return stablo;
    }

    public Forest<TreeNode, Veza> kreni() {
        return stablo;
    }
}
```

```
public ObjekatPrikaza naKraj() {  
    ...  
    return op;  
}  
  
public Forest<TreeNode, Veza> naPocetak() {  
    ...  
    return stablo;  
}  
  
public void sakrijSve() {  
    ...  
}  
  
...
```

U okviru klase, postoje sledeća polja:

- trenutniPrikaz čuva broj trenutnog koraka do kog je stigao algoritam. Početna vrednost -1 označava da algoritam još uvek nije počeo sa izvršavanjem.
- stablo je objekat tipa *DelegateTree<TreeNode, Veza>* i predstavlja stablo odlučivanja u formatu pogodnom za JUNG sistem.
- listaCvorova čuva listu referenci ka svim čvorovima koji postoje u stablu. Zahvaljujući ovoj listi, moguće je pristupiti svakom čvoru i po potrebi izmeniti neko od polja – npr. vidljivost tog čvora.

Takođe, ova klasa sadrži i mehanizam za kretanje kroz sam algoritam – to obavljaju metodi *kreni()*, *napred()*, *nazad()*, *naKraj()* i *naPocetak()*.

S obzirom da ovi metodi vraćaju objekat stabla – ili direktno, ili upakovano u objekat klase *ObjekatPrikaza*, postavljena je osnova komunikacije sa bibliotekom JUNG koja će prihvatiti stablo, a potom i izvršiti njegovo predstavljanje.

U kontekstu prikaza stabla, važne su i *Predicate* klase - *TreeLinkPredicate* i *TreeNodePredicate*. Ove klase odlučuju koji će se čvorovi i veze u određenom koraku prikazivati. Njihova struktura je slična, a listing 4 prikazuje klasu *TreeNodePredicate*.

Listing 4: Klasa *TreeNodePredicate*.

```
public class TreeNodePredicate implements
    Predicate<Context<Graph<ID3.TreeNode, ID3.Veza>, ID3.TreeNode>> {

    protected boolean showUnvisited;

    public TreeNodePredicate(boolean filter) {
        this.showUnvisited = filter;
    }

    public void setShowUnvisited(boolean b) {
        this.showUnvisited = b;
    }

    public boolean evaluate(Context<Graph<ID3.TreeNode, ID3.Veza>,
        ID3.TreeNode> context) {
        ID3.TreeNode v = (ID3.TreeNode)context.element;
        return v.isVisible();
    }
}
```

4.4.3. Implementacija klasterovanja

Osnovna klasa od koje modul klasterovanje počinje izvršavanje jeste *kmPanel*. U okviru ove klase poziva se metod *pokreniAlgoritam()*. Ovaj metod pokreće izvršenje algoritma. U svakom koraku se određuje koliko tačaka pripada kom klasteru i izračunava se tačna pozicija centroida. Listing 5 prikazuje klasu *Korak* koja čuva informacije o svakoj iteraciji algoritma.

Listing 5: Prikaz klase *Korak*.

```
public class Korak {
    Tacka[] centar;
    int[][] pripadnost;
    String poruka;

    public Korak() {
        centar = new Tacka[klasteriSlajder.getValue()];
        pripadnost = new int[klasteriSlajder.getValue()][];
        poruka = "";
    }
}
```

Najvažnija polja su:

- *centar* koji čuva poziciju centroida svakog klastera.
- *pripadnost* u okviru koje se čuvaju pozicije svih tačaka koje pripadaju određenom klasteru.

Za potrebe grafičkog prikazivanja koristi se paket *Java AWT*. Listing 6 prikazuje deo programske logike za iscrtavanje tačaka i klastera.

Listing 6: Deo programske logike za iscrtavanje tačaka i klastera.

```

if(prikaziIstoriju) {
    Ellipse2D krug;
    for(int m=0; m<k.centar.length; m++) {
        double x_s = koraci.get(0).centar[m].getX();
        double y_s = koraci.get(0).centar[m].getY();
        g2.setColor(proveriBoju(m));
        krug = new Ellipse2D.Double(x_s-KRUG_VELICINA/2+sirina,
        y_s-KRUG_VELICINA/2+visina, KRUG_VELICINA, KRUG_VELICINA);
        g2.fill(krug);
        g2.draw(krug);
        for(int n=1; n<=trenutniKorak; n++) {
            double x = koraci.get(n).centar[m].getX();
            double y = koraci.get(n).centar[m].getY();
            if(n==trenutniKorak) {
                g2.draw(new Line2D.Double(
                x_s-(KRSTIC_VELICINA/2)+sirina, y_s+visina,
                x+(KRSTIC_VELICINA/2)+sirina, y+visina));

                Rectangle2D kvadrat = new Rectangle2D.Double(
                x-KRUG_VELICINA/2+sirina,
                y-KRUG_VELICINA/2+visina, KRUG_VELICINA,
                KRUG_VELICINA);
                g2.setColor(Color.WHITE);
                g2.fill(kvadrat);
                g2.setColor(proveriBoju(m));
                g2.draw(kvadrat);
                x_s = x;
                y_s = y;
            } else {
                krug = new Ellipse2D.Double(
                x-KRUG_VELICINA/2+sirina,
                y-KRUG_VELICINA/2+visina,
                KRUG_VELICINA, KRUG_VELICINA);
                g2.fill(krug);
                g2.draw(krug);
                g2.draw(new Line2D.Double(
                x_s-(KRSTIC_VELICINA/2)+sirina, y_s+visina,
                x+(KRSTIC_VELICINA/2)+sirina, y+visina));
                x_s = x;
                y_s = y;
            }
        }
    }
}

```

```
} else {  
    for(int m=0; m<k.centar.length; m++) {  
        g2.setColor(proveriBoju(m));  
        double x = k.centar[m].getX();  
        double y = k.centar[m].getY();  
        Ellipse2D krug = new Ellipse2D.Double(x-KRUG_VELICINA/2+sirina,  
            y-KRUG_VELICINA/2+visina, KRUG_VELICINA, KRUG_VELICINA);  
        g2.fill(krug);  
        g2.draw(krug);  
    }  
}
```

Pomoću *if* naredbe se proverava da li je odabrana opcija za iscrtavanje istorije – na osnovu toga će zavisi i način iscrtavanja.

Bitno je naglasiti da je generisanje tačaka nasumično pomoću funkcije *random()*. Ali ono što je još važnije je to da je *seed* ove funkcije fiksiran, kao što je napomenuto u prethodnom delu teksta.

4.4.4. Implementacija algoritma Naive Bayes

Izvršavanje ovog modula se kontroliše pomoću klase *nbPanel*. Nakon što se unese ulazni skup podataka, iz posebne tabele se mogu izabrati test primeri koji se nalaze u odgovarajućim padajućim listama. Način na koji su padajuće liste povezane sa tabelom prikazan je u listingu 7.

Listing 7: Prikaz padajućih listi za izbor test primera.

```
ModelTabele mT = new ModelTabele();  
tabelaTestPrikaz = new JTable(mT);  
tabelaTestPrikaz.setRowHeight(20);  
tabelaTestPrikaz.setRowSelectionAllowed(false);  
TableColumnModel columnModel = tabelaTestPrikaz.getColumnModel();  
  
for(int i=0; i < naziviKolona.length; i++) {  
    JComboBox kb = new JComboBox();  
    System.out.println("Usla petlja");  
    for(int j = 0; j < me.domains[i].size(); j++) {  
        kb.addItem(me.domains[i].get(j));  
    }  
    TableColumn kolona = columnModel.getColumn(i);  
    kolona.setCellEditor(new DefaultCellEditor(kb));  
    me.testSet[0][i] = kb;  
}
```

Model tabele predstavlja zasebna klasa, *ModelTabele*. Zadužena je za sinhronizaciju podataka koji se nalaze u tabeli i u odgovarajućem nizu koji je zadužen za čuvanje tih vrednosti. Listing 8 prikazuje ovu klasu.

Listing 8: Prikaz klase *ModelTabele*.

```
class ModelTabele extends AbstractTableModel {
    public String getColumnName(int c) {
        String [] naziviKolona = new
            String[me.getAttributeNames().length];

        for(int i=0; i<me.getAttributeNames().length; i++) {
            naziviKolona[i] = me.getAttributeNames()[i];
        }
        return naziviKolona[c];
    }

    public int getColumnCount() {
        return me.getAttributeNames().length;
    }

    public int getRowCount() { return 1; }

    public Object getValueAt(int r, int c) {

        return me.izabraniUlazi[c];
    }

    public void setValueAt(Object obj, int r, int c) {
        me.izabraniUlazi[c] = (String) obj;
    }

    public boolean isCellEditable(int r, int c)
    {
        return true;
    }
}
```

Pošto se i ovde, kao i u slučaju stabla odlučivanja, generiše struktura podataka stablo, potrebno je napraviti odgovarajuće čvorove. U tu svrhu se koristi klasa *CvorNB* čija deo strukture je prikazan u listingu 9.

Listing 9: Deo strukture klase *CvorNB*.

```
public class CvorNB {
    private String porukaCvor;
    private double brojUkupnih;
    private double brojUslovnih;
    private boolean visible;
    private ArrayList<Integer> uslovnePozicije;
```

```
public CvorNB() {  
    porukaCvor = "";  
    brojUkupnih = 0.0;  
    brojUslovnih = 0.0;  
    visible = false;  
    uslovnePozicije = new ArrayList<Integer>();  
}  
...
```

Najvažnija polja su:

- *brojUkupnih* predstavlja ukupan broj vrednosti koje su ušle u obračun.
- *brojUslovnih* predstavlja deo ukupnih vrednosti koje ispunjavaju određeni uslov. Količnik uslovnih i ukupnih vrednosti daje uslovnu verovatnoću određenog atributa.
- *visible* kao i kod stabla odlučivanja čuva oznaku da li trenutni čvor treba prikazati ili ne.
- *uslovnePozicije* predstavljaju brojeve redova iz tabele koji se koriste u obračunu. Pozadinska boja redova koji su pobrojani u ovom polju biće promenjena kako bi se naznačilo da su trenutno u upotrebi.

4.4.5. Implementacija perceptrona

Radom ovog modula se upravlja pomoću klase *perceptronPanel*. Izvršenje algoritma počinje pozivanjem metoda *pokreniAlgoritam()*. Početak ovog metoda prikazan je u listingu 10.

Listing 10: Deo metoda *pokreniAlgoritam()*.

```
boolean greska = true;  
int trenutnaIteracija = 0;  
  
while(greska == true && trenutnaIteracija <= ukupnoIteracija) {  
    int brojTacnih = 0;  
    for(int i=0; i < ulaz.length; i++) {  
        Korak k = new Korak();  
        int izlaz;
```

```
double suma = w1*ulaz[i][0] + w2*ulaz[i][1];
if(suma >= teta) {
    izlaz = 1;
} else {
    izlaz = 0;
}
```

U ovom listingu prikazan je deo koda koji prikazuje način na koji perceptron obračunava vrednost. U promenljivu *suma* smešta se zbir proizvoda ulaznih vrednosti i težina. Na osnovu poređenja te sume sa pragom, određuje se izlazna vrednost. Pošto se radi o oštroj pragovskoj funkciji, izlazna vrednost može biti ili 0 ili 1.

Ukoliko se izlazna vrednost perceptrona ne poklapa sa traženom izlaznom vrednošću, potrebno je korigovati težine. To se postiže pomoću programskog koda prikazanom u listingu 11.

Listing 11: Programski kod za korigovanje težina.

```
if(izlaz == 1) {
    BigDecimal alfaB = new BigDecimal(""+alfa);
    BigDecimal ulazB1 = new BigDecimal(""+(double)ulaz[i][0]);
    BigDecimal ulazB2 = new BigDecimal(""+(double)ulaz[i][1]);
    BigDecimal w1B = new BigDecimal(""+w1);
    BigDecimal w2B = new BigDecimal(""+w2);
    w1B = w1B.subtract(alfaB.multiply(ulazB1));
    w1 = w1B.doubleValue();

    w2B = w2B.subtract(alfaB.multiply(ulazB2));
    w2 = w2B.doubleValue();

    k.setW1(w1);
    k.setW2(w2);
    k.setAlfa(alfa);
    k.setTeta(teta);
    k.setUvecavanje(false);
} else {
    BigDecimal alfaB = new BigDecimal(""+alfa);
    BigDecimal ulazB1 = new BigDecimal(""+(double)ulaz[i][0]);
    BigDecimal ulazB2 = new BigDecimal(""+(double)ulaz[i][1]);
    BigDecimal w1B = new BigDecimal(""+w1);
    BigDecimal w2B = new BigDecimal(""+w2);

    w1B = w1B.add(alfaB.multiply(ulazB1));
    w1 = w1B.doubleValue();

    w2B = w2B.add(alfaB.multiply(ulazB2));
    w2 = w2B.doubleValue();
}
```

```
k.setW1(w1);  
k.setW2(w2);  
k.setAlfa(alfa);  
k.setTeta(teta);  
k.setUvecavanje(true);  
}
```

Za obračun težina, ne koristi se tip podataka *double* zbog mogućeg gubitka preciznosti u radu sa malim brojevima. Koristi se tip *BigDecimal*, koji je pogodan zbog visoke preciznosti rada. Ukoliko je izlaz perceptrona 1, vrednosti težine će se smanjivati. U suprotnom, ukoliko je izlaz 0, težine će se uvećavati.

Kako bi grafička prezentacija perceptrona imala iste proporcije na monitoru svake veličine, na osnovu dimenzija okvira se izračunava na koji način će biti iscrtan perceptron. Listing 12 prikazuje kod koji obavlja ovaj posao.

Listing 12: Prilagođavanje grafičke predstave perceptrona.

```
Dimension d = getSize();  
sirina = d.width;  
visina = d.height;  
  
double segment = sirina/10;  
double segmentVisina = visina*0.3;
```

Na osnovu izračunatih vrednosti segmenata biće napravljen grafički prikaz perceptrona. Deo programskog koda zadužen za njegovo iscrtavanje prikazan je u listingu 13.

Listing 13: Prilagođavanje grafičke predstave perceptrona.

```
g2.setColor(Color.BLUE);  
g2.draw(new Line2D.Double(2*segment, segmentVisina,  
                           4*segment, segmentVisina));  
g2.draw(new Line2D.Double(2*segment, 2*segmentVisina,  
                           4*segment, 2*segmentVisina));  
  
g2.draw(new Line2D.Double(4*segment, segmentVisina,  
                           6*segment, visina*0.45));  
g2.draw(new Line2D.Double(4*segment, 2*segmentVisina,  
                           6*segment, visina*0.45));  
  
g2.draw(new Line2D.Double(6*segment, visina*0.45,  
                           8*segment, visina*0.45));
```

Na primeru iscrtavanja linija vidi se na koji način se primenjuju vrednosti izračunatih segmenata.

4.4.6. Implementacija algoritama pretraživanja

Implementacija modula za algoritme pretraživanja deli osnovu sa modulima za algoritme mašinskog učenja. Način organizacije interfejsa i raspored osnovnih klasa je urađen na sličan način tako da ti segmenti neće biti ponovo obrađivani. Zato će u ovoj sekciji biti obrađene samo specifičnosti implementacije algoritama pretraživanja.

S obzirom da je u okviru modula za algoritme pretraživanja osnovna struktura sa kojom se radi graf, u toku razvoja je bilo potrebno uložiti dosta pažnje u implementaciju ove strukture podataka. Kao velika pomoć u grafičkoj reprezentaciji grafova korišćena je biblioteka JUNG.

Prvi korak u radu sa algoritmima pretraživanja je pravljenje novog grafa. Za to je zadužena klasa *GraphEditor.java*. Listing 14 prikazuje deo programskog koda ove klase, tačnije deo konstruktora u okviru koga se inicijalizuje interfejs.

Listing 14: Deo konstruktora klase *GraphEditor.java*.

```
public GraphEditorDemo() {  
    ...  
  
    // Pravljenje novog grafa ili učitavanje postojećeg  
    ...  
  
    this.layout = new StaticLayout<GrafCvor,GrafVeza>(graph,  
        new Dimension(600,600));  
  
    vv = new VisualizationViewer<GrafCvor, GrafVeza>(layout);  
    vv.setBackground(Color.white);  
  
    ...  
  
    Container content = getContentPane();  
    final GraphZoomScrollPane panel = new GraphZoomScrollPane(vv);  
    content.add(panel);  
    ...  
  
    vv.setGraphMouse(graphMouse);  
    vv.addKeyListener(graphMouse.getModeKeyListener());  
  
    graphMouse.setMode(ModalGraphMouse.Mode.EDITING);  
}
```

```
final ScalingControl scaler = new CrossoverScalingControl();
JButton plus = new JButton("+");
plus.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        scaler.scale(vv, 1.1f, vv.getCenter());
    }
});
JButton minus = new JButton("-");
minus.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        scaler.scale(vv, 1/1.1f, vv.getCenter());
    }
});

JButton ok = new JButton("OK");
JButton cancel = new JButton("Poništi");

...

JPanel controls = new JPanel();
controls.add(plus);
controls.add(minus);
JComboBox modeBox = graphMouse.getModeComboBox();
controls.add(modeBox);

controls.add(ok);
controls.add(cancel);

content.add(controls, BorderLayout.SOUTH);
```

Prvo se pravi objekat novog grafa, ili se ukoliko u simulatoru već postoji aktivan graf, vrši njegovo učitavanje. Potom se definiše osnova rasporeda komponenti (promenljiva *layout*), a potom pravi i *VisualizationViewer* objekat (promenljiva *vv*) koji se zasniva na izabranom *layout*-u. Nakon toga se nastavlja sa definisanjem prikaza - prave se komande za uvećavanje i umanjivanje strane, dugmad za čuvanje i poništavanje napravljenih izmena i potom se dodaju u dijalog.

Graf se sastoji od elemenata tipa *GrafCvor* i *GrafVeza*, koji odgovaraju čvorovima i vezama. Listing 15 prikazuje početak klase *GrafCvor*.

Listing 15: Osnova klase GrafCvor.

```
public class GrafCvor
{
    private String naziv;
    private int id;
    private boolean trenutni;
    private boolean posecivan;
    private boolean pocetni;
    private boolean krajnji;
    private double heuristics;
    ...
}
```

Radi jednostavnosti su isključeni metodi koji vrše upis i čitanje ovih polja. Sličan je izgled i klase GrafVeza koja definiše veze između pojedinačnih čvorova grafa. Listing 16 prikazuje početak klase *GrafVeza*.

Listing 16: Osnova klase GrafVeza.

```
public class GrafVeza
{
    private String naziv;
    private int id;
    private GrafCvor start;
    private GrafCvor end;
    private boolean posecivan;
    private double price;
    ...
}
```

Pri iscrtavanju svakog čvora ili veze, vodi se računa o tome kojom će bojom biti prikazan. Na primer, početni čvor se uvek obeležava crvenom bojom, a krajnji narandžastom. Listing 17 prikazuje metod koji na jednostavan način rukuje izborom boja.

Listing 17: Određivanje boje određenog čvora.

```
public Paint odrediBoju(GrafCvor cvor) {
    if (node.jePosecivan()) {
        return Color.YELLOW;
    } else if (node.jeTrenutni()) {
        return Color.MAGENTA;
    } else if (node.jePocetni()) {
        return Color.RED;
    } else if (node.jeKrajnji()) {
        return Color.ORANGE;
    }
}
```

```
    return Color.LIGHT_GRAY;  
}
```

Nakon što je graf napravljen i prikazan na ekranu, na njega se primenjuje određeni algoritam pretraživanja. Kreće se od čvora koji je korisnik označio kao početni i traži se putanja do određiškog čvora. Ova procedura se razlikuje u zavisnosti od izabranog algoritma, a prate se koraci koji su definisani u poglavlju koje se bavi analizom algoritama pretraživanja.

U cilju bolje kontrole rasporeda komponenti korisničkog interfejsa, u osnovi je korišćena klasa *GridBagLayout*. Ova klasa nudi izuzetno velike mogućnosti za postavljanje i kontrolisanje rasporeda komponenti interfejsa. Ovaj menadžer rasporeda upotrebljen je ne samo za pretraživanje, već i za ostale algoritme u simulatoru - listing 18 prikazuje jedan primer njegove upotrebe.

Listing 18: Upotreba GridBagLayout menadžera za raspoređivanje komponenti interfejsa.

```
setLayout(new GridBagLayout());  
  
add(toolBar, new GBC(0, 0, 2, 1).setAnchor(GBC.NORTH).setFill(GBC.HORIZONTAL)  
    .setWeight(100, 0).setInsets(0));  
add(ulaz, new GBC(0, 1).setAnchor(GBC.WEST).setInsets(20, 40, 5, 5));  
add(poruke, new GBC(1, 1).setAnchor(GBC.WEST).setInsets(20, 20, 5, 5));  
add(scrollPaneTabela, new GBC(0, 2).setAnchor(GBC.CENTER).setWeight(100, 100)  
    .setInsets(5, 40, 0, 20).setFill(GBC.BOTH));  
add(scrollPane, new GBC(1, 2).setAnchor(GBC.CENTER).setWeight(100, 100)  
    .setInsets(5, 20, 0, 40).setFill(GBC.BOTH));  
add(brisanjePoruka, new GBC(1, 3).setAnchor(GBC.CENTER).setInsets(10));  
add(stablo, new GBC(0, 4, 2, 1).setAnchor(GBC.CENTER).setWeight(100, 100)  
    .setInsets(0, 20, 0, 20).setFill(GBC.BOTH));  
add(kontrole, new GBC(0, 5, 2, 1).setAnchor(GBC.CENTER).setInsets(10, 0, 0, 0)  
    .setFill(GBC.HORIZONTAL));
```

Kako bi se olakšao relativno složen rad sa ovim menadžerom rasporeda, upotrebljena je dodatna klasa, *GBC.java* [125], koja olakšava raspoređivanje komponenti. Listing 19 prikazuje jednostavnu strukturu ove klase.

Listing 19: Dodatna klasa GBC.java za rad sa GridBagLayout menadžerom.

```
public class GBC extends GridBagConstraints {

    public GBC(int gridx, int gridy) {
        this.gridx = gridx;
        this.gridy = gridy;
    }
    public GBC(int gridx, int gridy, int gridwidth, int gridheight) {
        this.gridx = gridx;
        this.gridy = gridy;
        this.gridwidth = gridwidth;
        this.gridheight = gridheight;
    }
    public GBC setAnchor(int anchor) {
        this.anchor = anchor;
        return this;
    }
    public GBC setFill(int fill) {
        this.fill = fill;
        return this;
    }
    public GBC setWeight(double weightx, double weighty) {
        this.weightx = weightx;
        this.weighty = weighty;
        return this;
    }
    public GBC setInsets(int distance) {
        this.insets = new Insets(distance, distance, distance, distance);
        return this;
    }
    public GBC setInsets(int top, int left, int bottom, int right) {
        this.insets = new Insets(top, left, bottom, right);
        return this;
    }
    public GBC setIpad(int ipadx, int ipady) {
        this.ipadx = ipadx;
        this.ipady = ipady;
        return this;
    }
}
```

5. LABORATORIJSKE VEŽBE ZASNOVANE NA SOFTVERSKOM SISTEMU

Kako bi se maksimalno iskoristili kapaciteti softverskog sistema za vizuelnu simulaciju algoritama inteligentnih sistema, potrebno je obezbediti odgovarajući skup laboratorijskih vežbi koje na najbolji način mogu demonstrirati rad izabranih algoritama.

Cilj je da se pomoću zanimljivih primera demonstriraju mogućnosti softverskog simulatora i da korisnici na najlakši način shvate kako određeni algoritam funkcioniše.

5.1. Demonstracija rada stabala odlučivanja

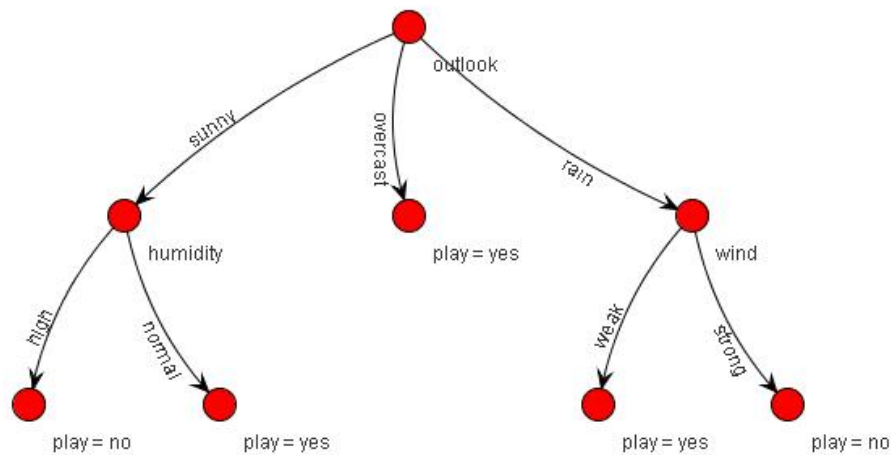
Kao jednostavniju demonstraciju od koje se može početi upoznavanje sa stablima odlučivanja, pogodno je upotrebiti *play tennis* [65] primer. Pomoću ovog primera, koji se u literaturi često koristi za osnovnu demonstraciju stabala odlučivanja, lako je prikazati način rada algoritma *ID3*. Ulazni skup podataka prikazan je u tabeli 1.

Tabela 1: Ulazni skup podataka primera *play tennis*.

Outlook	temp	humidity	wind	play
Sunny	hot	high	weak	no
Sunny	hot	high	strong	no
Overcast	hot	high	weak	yes
Rain	mild	high	weak	yes
Rain	cool	normal	weak	yes
Rain	cool	normal	strong	no
Overcast	cool	normal	strong	yes
sunny	mild	high	weak	no
sunny	cool	normal	weak	yes
rain	mild	normal	weak	yes
sunny	mild	normal	strong	yes
overcast	mild	high	strong	yes
overcast	hot	normal	weak	yes
rain	mild	high	strong	no

U prvom redu se nalaze nazivi atributa (*outlook*, *temp*, *humidity*, *wind* i *play*), a njihove vrednosti su prikazane u narednim redovima. Ove zapisi predstavljaju statističke podatke o odigranim mečevima u nekom sportu. S obzirom da postoji 14 redova, to znači da su sakupljeni podaci sa 14 različitih susreta. Cilj je napraviti stablo odlučivanja – njegov izgled prikazan je na grafikonu 73. Ovo stablo može pomoći prilikom sledećeg meča - ukoliko imamo podatke o vremenskoj prognozi za taj dan, možemo na osnovu uslova koji su postojali na prethodnim mečevima, a uz pomoć stabla odlučivanja, doneti pretpostavku da li će biti moguće igrati ili vremenski uslovi to neće dozvoliti.

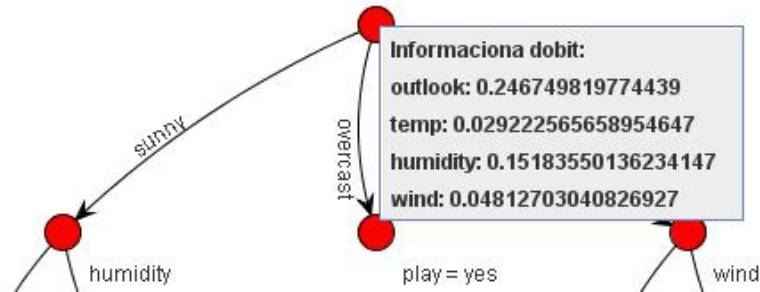
Grafikon 73: Stablo odlučivanja za primer *play tennis*.



Kao što se vidi na grafikonu, algoritam je isključio atribut *temp* smatrajući da nije važan za krajnju odluku. Upotrebljene su samo vrednosti *outlook*, *humidity* i *wind*.

Odluka o izboru najdiskriminativnijeg čvora donosi se na osnovu vrednosti entropije i informacione dobiti. Za početni čvor entropija je 0.9402859586706309 . Ukoliko se desnim tasterom klikne na njega (ili bilo koji drugi čvor) prikazaće se kontekstni meni sa informacionim dobitima za sve attribute koji nisu prethodno upotrebljeni (grafikon 74). To je ključna informacija na osnovu koje se vidi zbog čega je baš određeni atribut izabran. U navedenom primeru, atribut *outlook* ima najveću informacionu dobit što znači da je najpogodniji - zato je i izabran za koren stabla. Ista procedura se potom može ponoviti i za preostale attribute kako bi se proverile njihove entropije.

Grafikon 74: Informaciona dobiti za koreni čvor - outlook.



Na osnovu ovog test primera se vidi da će se meč igrati samo ako je vreme sunčano i vlažnost vazduha normalna, ako je oblačno ili ako je kišovito, pod uslovom da je vetar slab. Isto tako, meč se neće odigrati za slučaj da je vreme sunčano, a vlažnost velika ili da je kišovito uz snažan vetar.

Da bi se došlo do stabla odlučivanja, potrebno je napraviti statističko istraživanje pomoću koga će biti generisan skup ulaznih podataka. To se može postići i jednostavnim praćenjem i beleženjem događaja, kao u prethodnom primeru kada su praćeni vremenski uslovi. Kao primer se može uzeti i kupovina određenih usluga u zavisnosti od profila kupaca, što takođe može biti pogodno za razumevanje rada stabala odlučivanja. Naime, moguće je npr. ispred prodavnice anketirati osobe koje izlaze da li su kupile određeni proizvod ili uslugu i pritom zapisati osnovne podatke o njima (broj godina, obrazovanje, prihodi i bračni status). Za primer koji je naveden, taj skup izgleda kao u tabeli 2. Praćenjem ovakvih karakteristika može se zaključiti koja kategorija osoba bi najpre mogla da postane kupac. Na taj način se tržišni nastup može usmeriti ka tim kategorijama, dok se ulaganje resursa u ostale, koje ne predstavljaju ciljnu grupu, može redukovati.

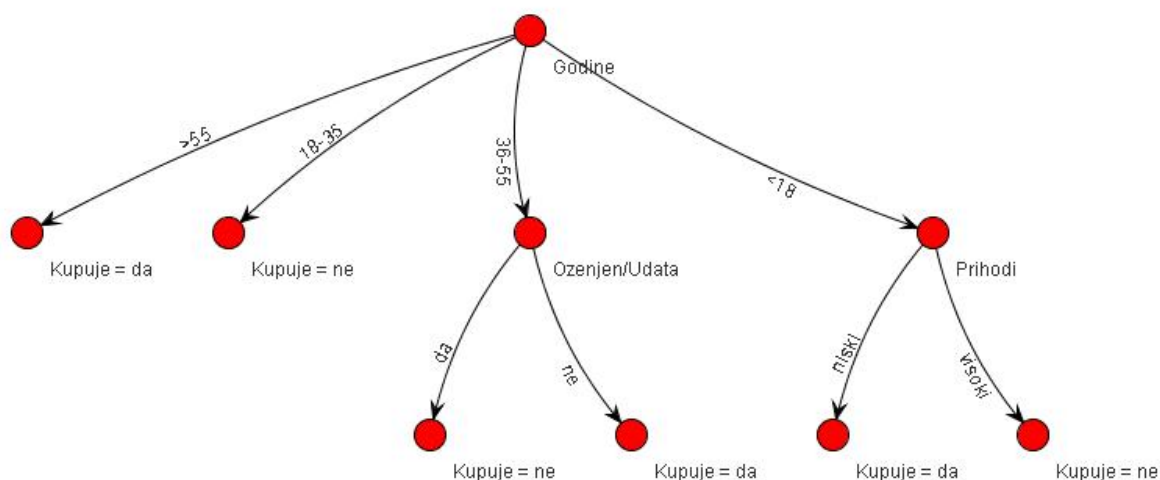
Tabela 2: Ulazni skup podataka za primer kupovine određenog proizvoda ili usluge.

Godine	Obrazovanje	Prihodi	Oženjen/Udata	Kupuje
36-55	VII	visoki	ne	da
18-35	V	niski	ne	ne
36-55	VII	niski	ne	da
18-35	VI	visoki	ne	ne
<18	V	niski	ne	da
18-35	VI	visoki	da	ne
36-55	VI	niski	da	ne
>55	VI	visoki	ne	da
36-55	VII	niski	da	ne
>55	VII	niski	da	da
36-55	VII	visoki	ne	da
>55	VII	visoki	ne	da
<18	V	visoki	ne	ne
36-55	VII	niski	ne	da
36-55	V	niski	ne	da
<18	V	niski	da	da
18-35	VI	visoki	da	ne
>55	V	visoki	da	da
>55	VI	niski	ne	da
36-55	V	visoki	da	ne

Prilagođeno prema: Roach C., *Building Decision Trees in Python*, O'Reilly Media [131]

Na osnovu vrednosti iz ulazne tabele, generiše se stablo odlučivanja prikazano na grafikonu 75.

Grafikon 75: Stablo odlučivanja za primer kupovine proizvoda ili usluga.



Iz navedenog primera se vidi da će osobe koje imaju preko 55 godina veoma verovatno biti kupci. Isto važi i za one starosti od 36-55 godina, ukoliko nisu u braku, kao i za mlađe od 18 godina u slučaju da imaju niske prihode. U ostalim slučajevima postoji velika verovatnoća da neće postojati interesovanje za uslugama.

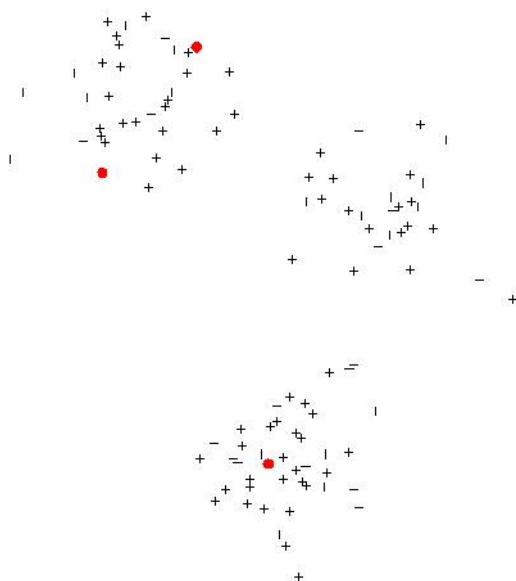
Ukoliko se prethodni grafikon pažljivije pogleda i uporedi sa odgovarajućom tabelom ulaznih podataka, primetiće se da je u finalnom stablu izostavljen jedan atribut (kao što je jedan atribut izostavljen i u *playtennis* slučaju) - *obrazovanje*. Algoritam je doneo odluku da ovaj faktor nije presudan za krajnji rezultat (tj. za odluku da li će kupovina biti napravljena) i zbog toga ga je isključio. To predstavlja odličan primer na koji način ovaj vid pomoći u poslovnom odlučivanju može biti od izuzetne koristi. Postoji velika verovatnoća da i stručnjak koji bi tumačio statističke podatke, ne bi mogao da predvidi jednu takvu zakonitost, pogotovo ako se radi o velikom skupu podataka.

5.2. Demonstracija rada klasterovanja

Kao što je već naglašeno, generisanje početnih tačaka za potrebe klasterovanja je nasumično, ali će zbog fiksiranog *seed*-a uvek za iste ulazne parametre broja tačaka, klastera i raspršenosti davati potpuno iste pozicije tačaka. To je dobro rešenje iz razloga jer je studentima na času moguće saopštiti koje vrednosti za tačke, klastere i raspršenost da unesu – potom će svi imati potpuno identičan prikaz na svojim računarima. Na taj način, kada svi na svojim ekranima imaju identične situacije, mnogo je lakše diskutovati o načinu rada algoritma.

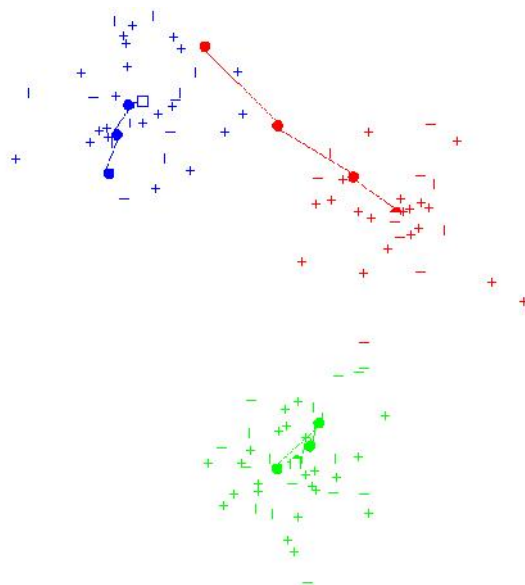
Kada se unese 110 tačaka i tri klastera pri raspršenosti od 5, dobija se raspored kao na grafikonu 76.

Grafikon 76: Početni prikaz tačaka za tri klastera.



Može se primetiti da su tačke približno grupisane u tri celine (klastera) – pritom centroidi su postavljeni nasumično na radnoj površini. Bilo bi logično očekivati da algoritam preraspodeli centroide i tačke koje im pripadaju tako da svakoj od tih grupa sa grafikona odgovara po jedan centroid. Za ovakvu preraspodelu biće potrebno nekoliko koraka – grafikon 77 prikazuje istoriju kretanja centroida i na koji način su dodeljivane tačke određenim klasterima.

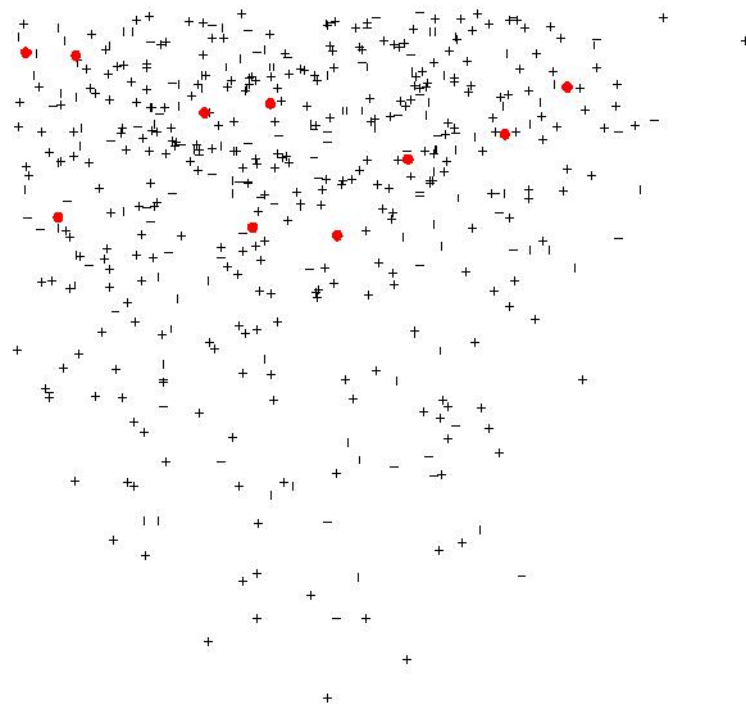
Grafikon 77: Prikaz tačaka i centroida na kraju rada algoritma.



Kao što se i vidi sa grafikona, centriodi su postavljeni baš onako kako je i bilo očekivano – po jedan u svaki od tri klastera.

Naravno, nisu uvek sve tačke grupisane kao u prethodnom primeru tako da se odmah može videti gde bi okvirno trebalo postaviti centroide. Postoje i složeniji slučajevi kada jedino sam algoritam može napraviti tačnu podelu. Ukoliko odaberemo maksimalne vrednosti za sve ulazne parametre – 500 tačaka, 10 klastera i maksimalna raspršenost – dobija se prikaz kao na grafikonu 78.

Grafikon 78: Početni prikaz tačaka za deset klastera.



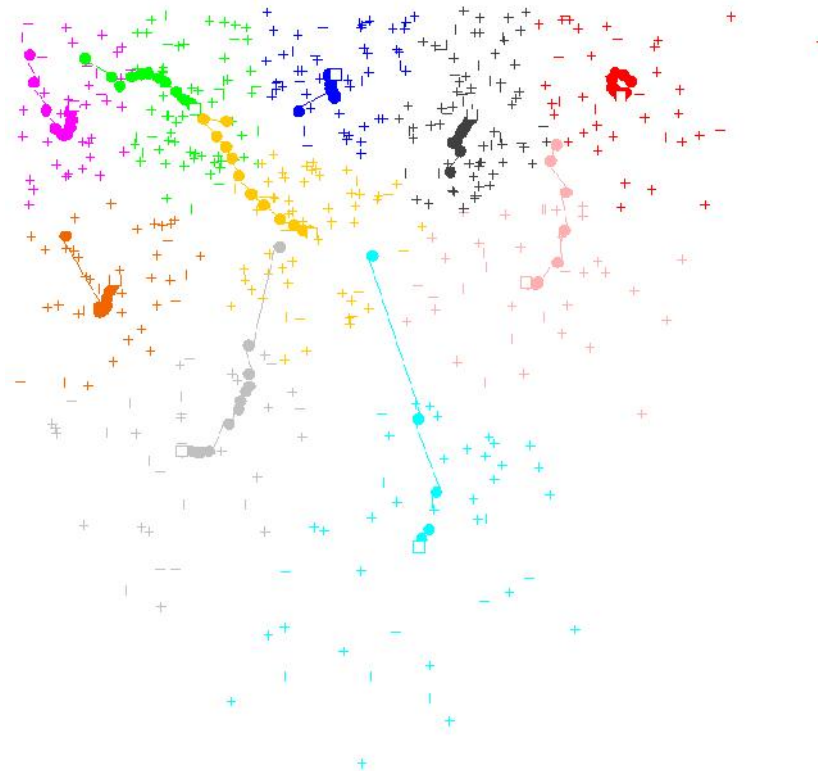
U ovom slučaju, jedino algoritam može naći krajnje rešenje i za to će mu biti potrebno 15 koraka. Tekstualni izlaz algoritma prikazan je u listingu 20. Prikazano je koliko je tačaka u svakom koraku promenilo centroid u kome se nalazi.

Listing 20: Tekstualni izlaz algoritma.

Potreban broj koraka: 15
Korak 1 - Početak algoritma
Korak 2 - Broj pomerenih tačaka: 90
Korak 3 - Broj pomerenih tačaka: 46
Korak 4 - Broj pomerenih tačaka: 28
Korak 5 - Broj pomerenih tačaka: 26
Korak 6 - Broj pomerenih tačaka: 24
Korak 7 - Broj pomerenih tačaka: 18
Korak 8 - Broj pomerenih tačaka: 25
Korak 9 - Broj pomerenih tačaka: 24
Korak 10 - Broj pomerenih tačaka: 14
Korak 11 - Broj pomerenih tačaka: 6
Korak 12 - Broj pomerenih tačaka: 3
Korak 13 - Broj pomerenih tačaka: 4
Korak 14 - Broj pomerenih tačaka: 1
Korak 15 - Broj pomerenih tačaka: 0
Nađeno rešenje!

Krajnji prikaz dat je na grafikonu 79. Jasno se vidi na koji način su grupisane tačke i kako su centriodi menjali mesta.

Grafikon 79: Prikaz tačaka i centroida na kraju rada algoritma.



5.3. Demonstracija rada Naive Bayes-a

Za potrebe učenja algoritma Naive Bayes može se krenuti od istog primera kao i za stablo odlučivanja – *play tennis* (tabela 1). Nakon što se vrednosti ovog primera učitaju u simulator, potrebno je izabrati vrednosti atributa za test primer (grafikon 80).

Grafikon 80: Izbor vrednosti test atributa.

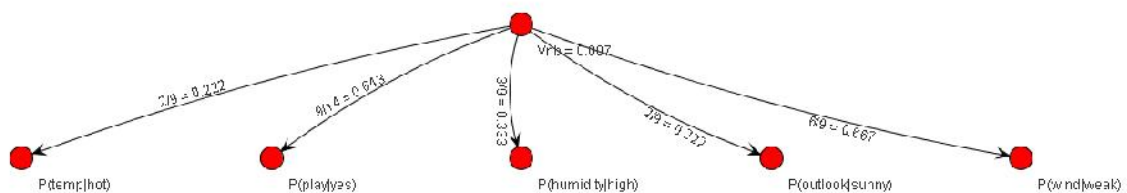
Test primer:

outlook	temp	humidity	wind	play
sunny	hot	high	weak	yes
				no
				yes

Algoritam radi tako što se za određenu kombinaciju atributa (u ovom slučaju vremenskih uslova) obračunava verovatnoća da li će se meč odigrati ili neće. Kao što se je prikazano na prethodnom grafikonu, u ovom slučaju je potrebno odrediti verovatnoću da će se teniski meč igrati, ukoliko je sunčano vreme, visoka temperatura i vlažnost vazduha, a vetar slab.

Nakon pokretanja algoritma, iscrtava se stablo sa prikazanim vrednostima uslovnih verovatnoća (grafikon 81).

Grafikon 81: Prikaz stabla sa vrednostima uslovnih verovatnoća.



U listovima stabla, nalaze se vrednosti uslovnih verovatnoća, kao i objašnjenje načina na koji su dobijene, tako da student može samostalno proveriti kako se došlo do određenih vrednosti. U korenu stabla nalazi se tražena verovatnoća da će meč biti odigran, uz zadate vremenske uslove iz test primera.

U okviru listinga 21 prikazan je i tekstualni izlaz programa pomoću koga se može videti kako algoritam funkcioniše.

Listing 21: Tekstualni izlaz algoritma Naive Bayes.

```

Korak 1
  P(play|yes) = 9 / 14 = 0.643
Korak 2
  P(outlook|sunny) = 2 / 9 = 0.222
Korak 3
  P(temp|hot) = 2 / 9 = 0.222
Korak 4
  P(humidity|high) = 3 / 9 = 0.333
Korak 5
  P(wind|weak) = 6 / 9 = 0.667
Korak 6
  Pronađeno rešenje
  Vnb = 0.007

```

Kao što se vidi iz korena stabla i tekstualnog izlaza algoritma, verovatnoća da će se igrati teniski meč je *0.007*. Da bi ova vrednost bila još korisnija, potrebno je izračunati verovatnoću da se pri zadatim vremenskim uslovima teniski meč neće odigrati. Ukoliko se ponovi prethodni postupak i za taj slučaj, dobiće se verovatnoća od *0.027*. Time zaključujemo da je veća verovatnoća da se teniski meč *neće* odigrati.

5.4. Demonstracija rada perceptrona

Kao pogodan primer za početak rada sa perceptronima mogu se koristiti logičke funkcije - u ovom slučaju izabrana je funkcija *AND*. Ona je izuzetno pogodna za demonstraciju rada algoritma jer je linearno separabilna, tako da rešenje postoji i algoritam će biti u mogućnosti da ga pronađe. Grafikon 82 prikazuje vrednosti ulaznih podataka.

Grafikon 82: Ulazni podaci za perceptron.

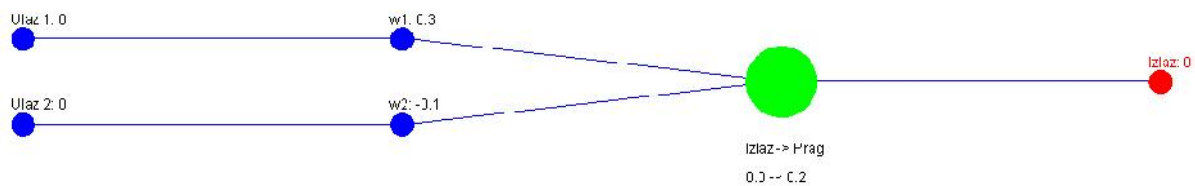
Test podaci:			
	Ulaz 1	Ulaz 2	Izlaz
	0	0	0
	0	1	0
	1	0	0
	1	1	1

Za početne parametre algoritma podešene su sledeće vrednosti:

- $w_1 = 0.3$;
- $w_2 = -0.1$;
- Stepen učenja – 0.1 ;
- Prag – 0.2 ;
- Maksimalni broj iteracija – 20 ;

Nakon pokretanja algoritma, perceptron u svakom koraku uzima po jedan red iz tabele sa ulaznim podacima i izračunava izlaz. Grafikon 83 prikazuje izlaz perceptrona za prvi korak algoritma.

Grafikon 83: Izlaz perceptrona za prvi korak algoritma.



Kao što se vidi, ulazi su 0 i 1, što i jesu vrednosti koje se nalaze u prvom primeru u tabeli sa ulaznim vrednostima. Takođe, uzimaju se i vrednosti koje su postavljene kao inicijalne težine, nakon čega se izračunava suma. Ako je suma veća od postavljene vrednosti praga, perceptron će vratiti vrednost 1. U suprotnom, izlaz će biti 0. Konkretno, u ovom slučaju, izlaz perceptrona će biti 0 jer je suma 0.0, a vrednost praga 0.2.

Na osnovu vrednosti izlaza perceptrona algoritam odlučuje da li je potrebno povećavati ili smanjivati veličine. Ukoliko se izlaz perceptrona poklapa sa izlazom koji se nalazi u ulaznoj tabeli, nema potrebe za korekcijom težina. U suprotnom, potrebno je izmeniti težine. Tekstualne poruke koje algoritam ispisuje prikazane su u listingu 22.

Listing 22: Tekstualni izlaz primera za logičku funkciju AND.

```
Iteracija 1 - korak 1:
  Izlaz perceptrona: 0 - Tačan izlaz: 0
  Izlaz se poklapa - nastavak algoritma.
Iteracija 1 - korak 2:
  Izlaz perceptrona: 0 - Tačan izlaz: 0
  Izlaz se poklapa - nastavak algoritma.
Iteracija 1 - korak 3:
  Izlaz perceptrona: 1 - Tačan izlaz: 0
  Izlaz se ne poklapa - izlazna vrednost perceptrona je 1 - vrši se umanjivanje.
  w1 = 0.3 - 0.1*1 = 0.2
  w2 = -0.1 - 0.1*0 = -0.1
Iteracija 1 - korak 4:
  Izlaz perceptrona: 0 - Tačan izlaz: 1
  Izlaz se ne poklapa - izlazna vrednost perceptrona je 0 - vrši se uvećavanje.
  w1 = 0.2 + 0.1*1 = 0.3
  w2 = -0.1 + 0.1*1 = 0.0
...

Iteracija 5 - korak 1:
  Izlaz perceptrona: 0 - Tačan izlaz: 0
  Izlaz se poklapa - nastavak algoritma.
Iteracija 5 - korak 2:
  Izlaz perceptrona: 0 - Tačan izlaz: 0
  Izlaz se poklapa - nastavak algoritma.
Iteracija 5 - korak 3:
  Izlaz perceptrona: 0 - Tačan izlaz: 0
  Izlaz se poklapa - nastavak algoritma.
Iteracija 5 - korak 4:
  Izlaz perceptrona: 1 - Tačan izlaz: 1
  Izlaz se poklapa - nastavak algoritma.
Nađeno rešenje!
```

Kao što se vidi iz izlaza algoritma, u prvom i drugom koraku prve iteracije nema nikakvih promena težine jer se vrednosti izlaza poklapaju. U trećem koraku iste iteracije, nije došlo do poklapanja tako da se vrši izmena težina, i to umanjivanjem jer je izlaz perceptrona 1. U četvrtom koraku takođe nema poklapanja, pa se vrši uvećavanje težina jer je izlaz perceptrona 0. Po ovakvom principu, algoritam nastavlja rad sve dok se u svakom koraku jedne iteracije ne dođe do tačnih rešenja. Za primer funkcije *AND*, do rešenja se stiglo u petoj iteraciji. Kao što se vidi iz prethodnog listinga, sva četiri koraka su dala tačno rešenje.

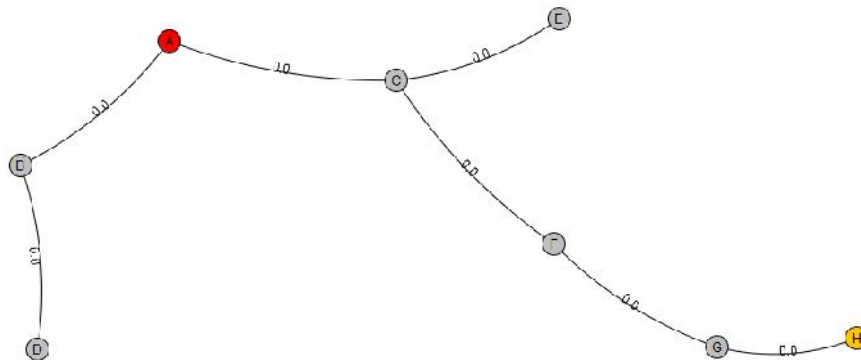
Stepen učenja koji je u ovom primeru podešen na 0,1 određuje brzinu, ali i preciznost algoritma. Praktično, što je niža vrednost stepena učenja, algoritam će praviti manje i

preciznije korake, pa će mu za dolazak do rešenja trebati više koraka, a samim tim i vremena. Ukoliko se stepen učenja poveća, krajnje rešenje će biti brže nađeno, ali se može desiti da njegova vrednost bude manje precizna. Ukoliko se do rešenja ne dođe u okviru definisanog maksimalnog broja koraka (u ovom slučaju 20), algoritam će prekinuti rad. Na taj način, obezbeđen je sigurnosni mehanizam koji prekida rad algoritma, u slučaju da rešenje ne postoji (skup koji nije linearno separabilan).

5.5. Demonstracija rada pretrage po širini

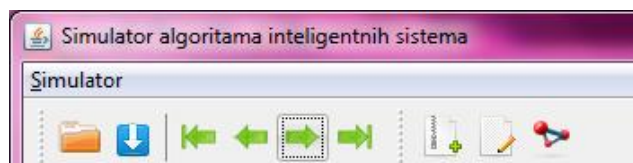
Kao početni primer za demonstraciju pretrage napravljeno je stablo kod koga se rešenje nalazi na nešto većoj dubini. Stablo je odabrano kao početni primer zbog lakšeg grafičkog prećenja rada algoritma pretrage. Za složenije primere se potom mogu koristiti i složeniji grafovi. Grafikon 84 prikazuje početno stanje. Početni čvor nosi oznaku A i prikazan je crvenom bojom, dok je krajnji čvor prikazan narandžastom bojom i ima naziv H.

Grafikon 84: Početni primer za demonstriranje rada pretrage u širinu.



Nakon što je izabran željeni algoritam pretraživanja (u ovom slučaju pretraga u širinu) opcije za kretanje kroz algoritam postaju dostupne - grafikon 85.

Grafikon 85: Aktivirane opcije za kretanje kroz algoritam.



Nakon što se počne sa kretanjem unapred kroz algoritam, na dnu ekrana će početi da se prikazuje stablo pretrage. Grafikon 86 prikazuje rezultat prvog koraka. Kao što se vidi na slici, iz početnog čvora A otkrivena su još dva čvora - B i C. Pošto B ima manju vrednost, u sledećem koraku će biti prvo obrađena ova grana.

Grafikon 86: Stablo pretrage nakon prvog koraka pretrage u širinu.



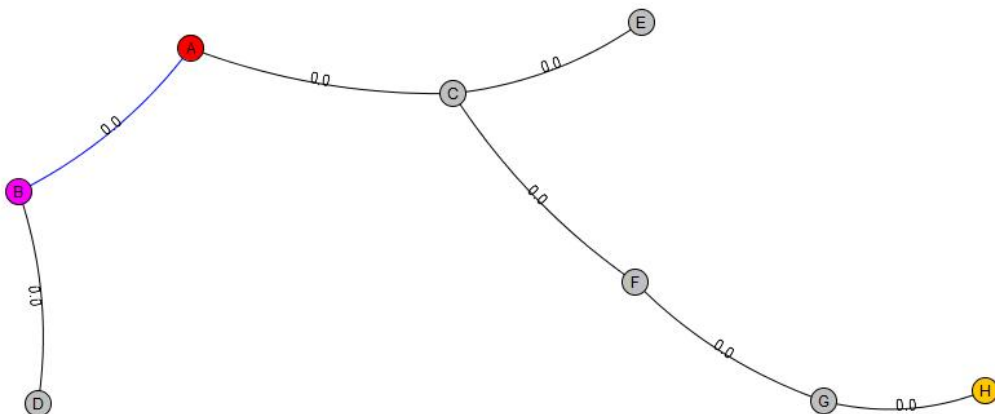
Ponovnim pomeranjem još jedan korak unapred prelazi se u čvor B - grafikon 87. Iz ovog čvora dostupan je samo jedan čvor - D.

Grafikon 87: Stablo pretrage nakon drugog koraka pretrage u širinu.



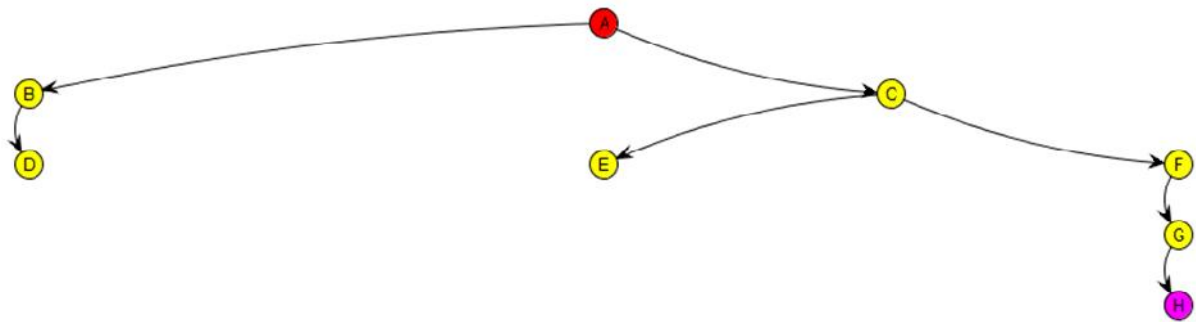
Paralelno sa iscrtavanjem stabla odlučivanja, u samom početnom prikazu se označavaju grane koje su trenutno aktivne - grafikon 88.

Grafikon 88: Početni prikaz i trenutno aktivna putanja.



Postepenim napretkom kroz algoritam, stiže se do krajnjeg rešenja - grafikon 89. Paralelno sa radom algoritma, u okviru za tekstualni ispis je prikazivan napredak u radu - posećeni i otkriveni čvorovi - u suštini, sve ono što je već prikazano u stablu pretrage. Zbog toga, dijalog za poruke u modulu za pretraživanje nije toliko važan kao u ostalim modulima simulatora.

Grafikon 89: Krajnji prikaz stabla pretraživanja.

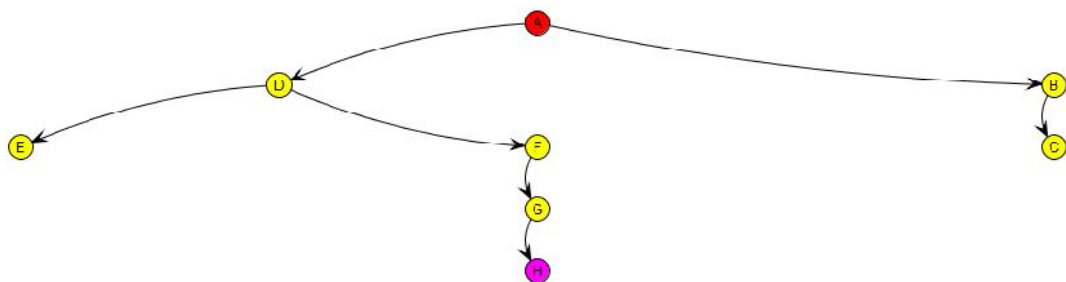


5.6. Demonstracija rada pretrage po dubini

I za demonstraciju pretrage po dubini upotrebljen je isti primer kao i za pretragu po širini - grafikon 84. Ovaj primer je pogodan zbog toga što se rešenje nalazi na većoj dubini, tako da će algoritmu trebati veći broj koraka da stigne do njega. Takođe, ovaj primer dobro pokazuje i problem koji pretraga po širini ima sa stablima kod kojih se rešenje nalazi na velikom rastojanju od početka.

S obzirom na sličnost sa pretragom po širini, za ovaj primer neće biti prikazivana međustanja, već samo krajnje rešenje - grafikon 90.

Grafikon 90: Krajnji rezultat pretrage po dubini.

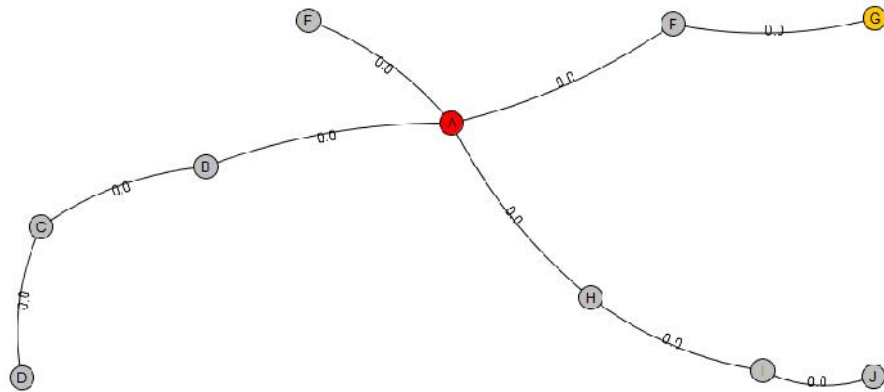


Kao što se i vidi, algoritam je stigao do rešenja nakon većeg broja koraka. Praktično, rešenje je bio poslednji posećeni čvor.

5.7. Demonstracija rada planinarenja

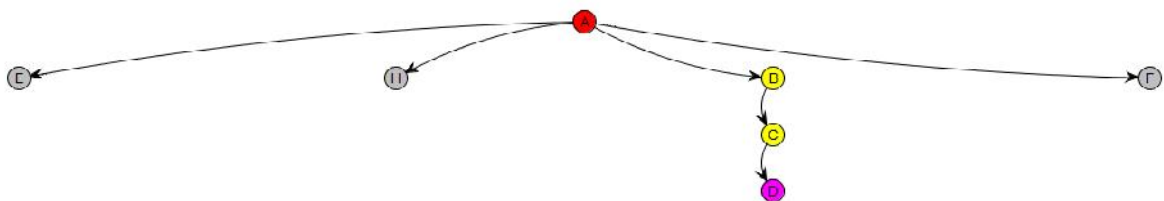
Za demonstraciju rada algoritma planinarenje izabran je primer u okviru koga, gledano od početnog čvora, postoji nekoliko grana, a u okviru samo jedne od njih se nalazi rešenje (grafikon 91). Na taj način, algoritam se već u prvom koraku podstiče da napravi izbor grane u okviru koje će početi da traži rešenje.

Grafikon 91: Primer za demonstraciju rada algoritma planinarenje.



Znajući osobinu ovog algoritma da se oslanja na slučajnu pretpostavku pozicije rešenja, velika je verovatnoća da iz prvog (ili nekoliko prvih) pokušaja neće krenuti pravim putem - grafikon 92. U ovom slučaju, algoritam je prvo krenuo da traži rešenje u pogrešnoj grani, nakon čega ju je u potpunosti istražio - tek nakon što se utvrdi da u okviru nje ne postoji rešenje, prelazi se na sledeću slučajno izabranu granu.

Grafikon 92: Simulacija rada algoritma planinarenje.

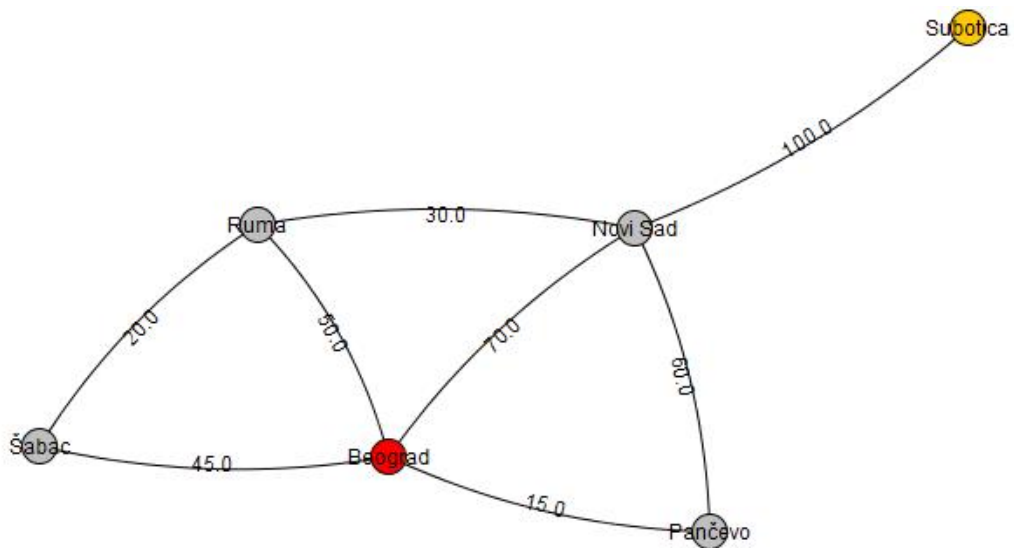


U ovom primeru, algoritam je tek iz trećeg puta uspeo da uđe u pravu granu i da njenim istraživanjem nađe rešenje.

5.8. Demonstracija rada algoritma A*

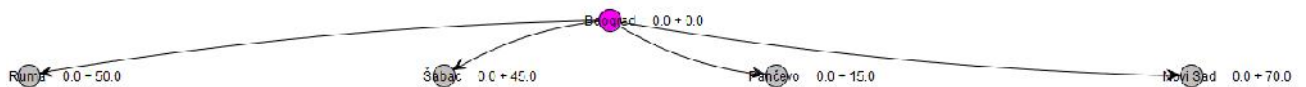
U odnosu na prethodno prikazane algoritme, A* je specifičan po tome što uključuje i cenu koštanja puta između čvorova. Zbog toga se može napraviti logička veza cene koštanja sa putnom mapom i kilometražama između gradova. Na grafikonu 93 je prikazan primer koji predstavlja mapu nekoliko gradova u Srbiji. Potrebno je iz Beograda što kraćim putem doći do Subotice. Očigledno je da postoji nekoliko mogućnosti od kojih je samo jedna optimalna.

Grafikon 93: Primer za demonstraciju rada algoritma A*.



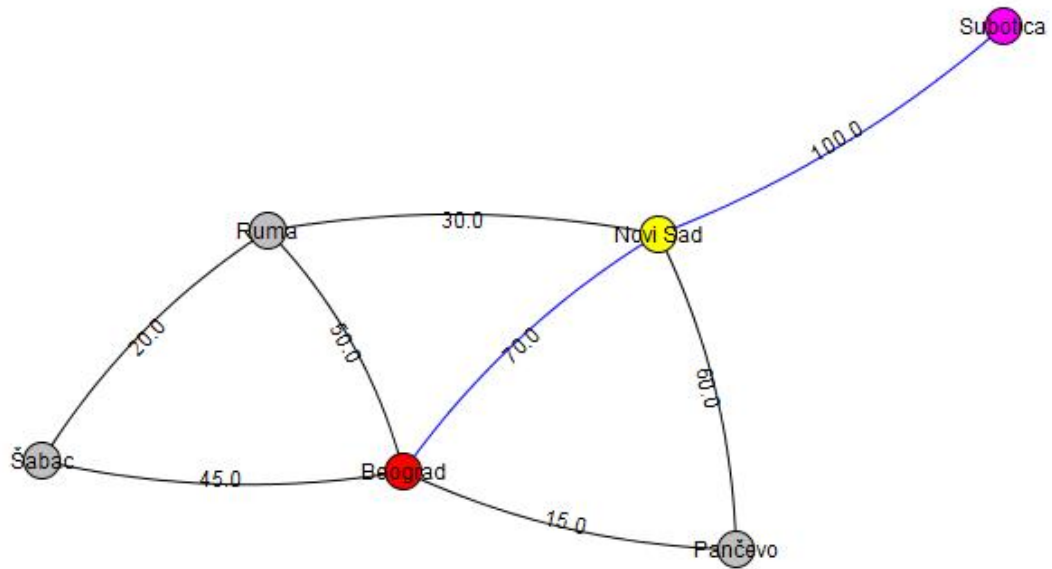
S obzirom da je A* heuristički algoritam, izuzetno puno se oslanja na cene koštanja. Na osnovu njih se određuje koji će čvor biti sledeći otvoren. Grafikon 94 prikazuje prvi korak algoritma za navedeni primer. Pored naziva svakog čvora, prikazuju se dve vrednosti - ukupna cena koštanja i cena koštanja između dva poslednja čvora. Za primer sa gradovima te vrednosti će predstavljati ukupnu kilometražu do trenutno izabranog čvora (grada na mapi) i kilometražu između trenutnog i prethodnog čvora.

Grafikon 94: Simulacija rada algoritma A*.



Algoritam će obračunati i ispitati razne mogućnosti, ali će, kao što je prikazano na grafikonu 95, kao krajnje rešenje izabrati zaista najkraći put.

Grafikon 95: Krajnje rešenje u radu algoritma A*.



6. EVALUACIJA

U okviru ovog poglavlja sumira se upotrebnost vrednost novog simulatora koji je predstavljen u ovom radu. Zahvaljujući procesu evaluacije, moguće je utvrditi da li je korišćenje ovog softvera doprinelo pozitivnim promenama na predmetu na kom se koristi, u smislu prolaznosti, prosečnih ocena, kao i studentskog zadovoljstva.

Simulator se koristi na Poslovnom fakultetu Valjevo Univerziteta Singidunum u okviru laboratorijskih vežbi iz predmeta Inteligentni informacioni sistemi. Predmet već nekoliko godina čini sastavni deo kurikuluma, a od toga, poslednje četiri godine je u upotrebi simulator inteligentnih sistema.

Upotrebom softverskog simulatora za učenje inteligentnih sistema koriste imaju i studenti i njihovi nastavnici. Prednosti za studente se ogledaju u nekoliko numeričkih indikatora, kao što je procenat studenata koji su položili ispit, njihove prosečne ocene, kao i broj studenata koji je izabrao izborni predmet u okviru koga se inteligentni sistemi uče. Prednosti za nastavnike se ogledaju u tome što se određeni teorijski koncepti mogu objasniti korišćenjem praktičnih primera i zadavanja interaktivnih zadataka. U ovom poglavlju će u cilju verifikacije navedenih prednosti biti obrazloženi skupovi podataka koji su korišćeni kao uzorak, analiza rezultata i diskusija.

6.1. Uzorak podataka

Kako bi se evaluirala upotreba sistema sprovedena je kvalitativna i kvantitativna analiza. Proces evaluacije je obuhvatio uzorke podataka u periodu posmatranja između 2007. i 2013. godine. U školskim godinama 2007/2008., 2008/2009. i 2009/2010. simulator za laboratorijske vežbe nije postojao. U toku školske 2010/2011. godine prva verzija simulatora je razvijena i uključena u korišćenje na laboratorijskim vežbama.

Na početku i kraju svake školske godine, na Poslovnom fakultetu Valjevo se sprovodi kvalitativna evaluacija za svaki predmet. Ova evaluacija je opciono i anonimno istraživanje u okviru koga svi studenti koji su pratili predmet popunjavaju anketu. Odgovaranjem na pitanja, oni evaluiraju kvalitet predmeta koji su slušali koristeći ocene od 5 do 10. Uzorak koji je uzima učešće u istraživanju čini grupa od oko 150 studenata godišnje.

Za kvantitativnu analizu na predmetu Inteligentni informacioni sistemi u toku posmatranog perioda su korišćeni statistički metodi, a posmatrane su dve grupe studenata. Članovi *eksperimentalne grupe* grupe su pohađali laboratorijske vežbe, za razliku od članova iz *kontrolne grupe* koji nisu. Nastavnici nisu formirali grupe - studenti su samostalno odlučivali kojoj grupi žele da pristupe. Važno je naglasiti da nisu bili svesni činjenice da će njihov napredak biti praćen i analiziran.

6.2. Analiza podataka

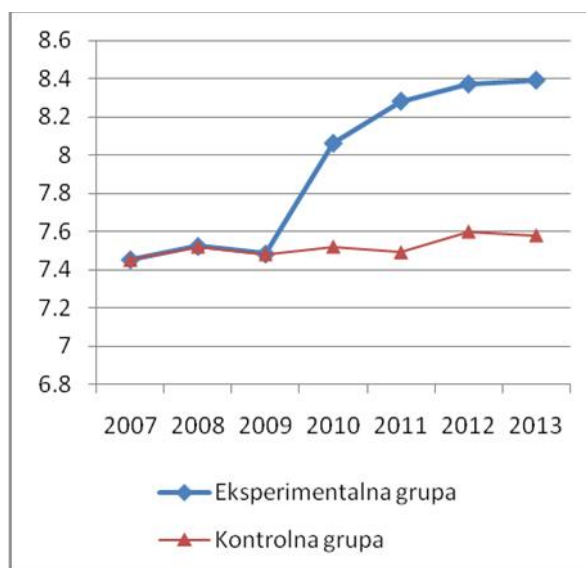
Analiza podataka izvršene ankete pokazuje da je prosečna ocena na predmetu Inteligentni informacioni sistemi porasla sa 8.1 na 8.9 na početku, kao i sa 8.9 na 9.4 na kraju školske godine (tabela 3). Pri tom, teme koje se obrađuju i nastavni materijali na predmetu se nisu menjali tokom posmatranog perioda. Prosečne ocene na drugim kursevima istog nivoa se nisu toliko drastično menjale tokom posmatranog perioda. Prosečna ocena koju je simulator dobio u prvom anketiranju je bila niža nego u drugom.

Tabela 3: Rezultati istraživanja.

	2010.	2011.	2012.	2013.
Prvo anketiranje	8.1	8.3	8.8	8.9
Drugo anketiranje	8.9	9.2	9.3	9.4

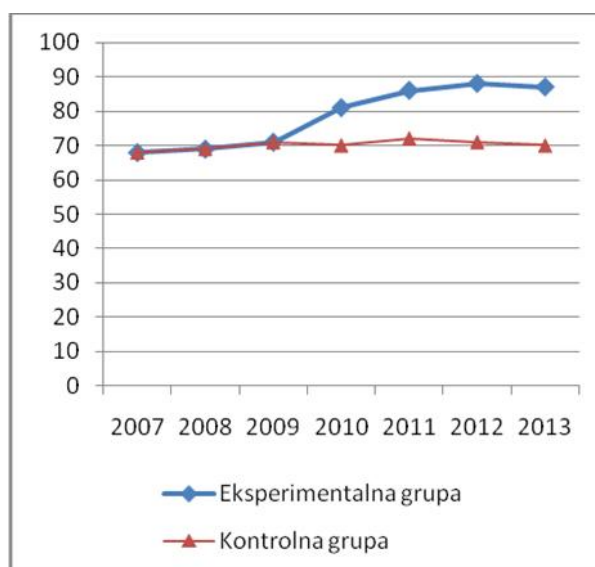
Kvantitativna analiza podataka je bazirana na dva numerička pokazatelja - to su prosečna ocena i procenat studenata koji su položili ispit. Grafikon 96 kroz prosečne ocene prikazuje uspeh studenata na predmetu u posmatranom periodu. Ocene su gradirane između 5 i 10, gde je ocena 5 najmanja, a 10 najviša. I eksperimentalna i kontrolna grupa su prikazane u okviru istog grafikona.

Grafikon 96: Prosečna ocena na predmetu Inteligentni informacijski sistemi.



Grafikon 97 prikazuje uspeh koji su postigli studenti koji su učestvovali u laboratorijskim vežbama. Poredeći školske godine 2010/2011, 2011/2012, 2012/2013 i 2013/2014 može se zapaziti da se svake godine povećavao broj studenata koji su prešli prag od 50% i postigli više ocene. Još jedan značajan indikator je i povećanje broja studenata koji su položili ispit u prvom ispitnom roku.

Grafikon 97: Procenat studenata koji su položili ispit.

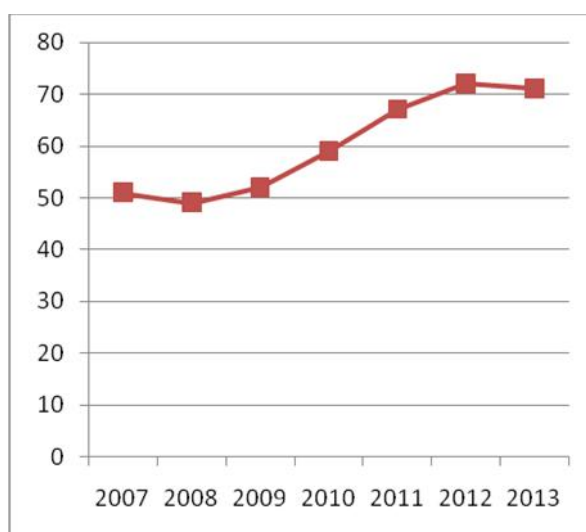


Za analizu studentskog uspeha primenjen je dvosmerni ANOVA test. Prvi faktor u analizi je bila posmatrana godina sa četiri nivoa (2010, 2011, 2012 i 2013). Drugi faktor je bila posmatrana grupa sa dva nivoa (eksperimentalna i kontrolna). Zavisne promenljive su bile studentske ocene za posmatranu grupu u toku posmatrane godine.

Glavni efekat faktora posmatrane grupe je značajan što znači da postoji razlika između eksperimentalnih i kontrolnih grupa. Glavni efekat faktora posmatrane godine nije statistički značajan što znači da ne postoji povećanje u stečenom znanju kod studenata između godina. Objašnjenje se može naći u činjenici da se nakon određenog broja naučenih algoritama dostiže zasićenje informacijama. Ni međusobni uticaj između ovih faktora nije statistički značajan što znači da razlika između eksperimentalne i kontrolne grupe nije zavisna u odnosu na godinu posmatranja.

Predmet Inteligentni informacioni sistemi se nudi kao izborni predmet na svim smerovima na Poslovnom fakultetu Valjevo. Pre uvođenja novog simulatora, 30% studenata je biralo ovaj kurs (grafikon 98): 51 od 150 u 2007., 49 od 148 u 2008. i 52 od 150 u 2010. godini. Nakon uključivanja simulatora u nastavu 2010. godine, broj studenata koji biraju ovaj kurs prikazuje konstantan porast (59 od 150 u 2010., 67 od 148 u 2011., 74 od 147 u 2012. i 71 u 2013.). Broj studenata koji su izabrali ovaj predmet je 2012. prvi put premašio 50% ukupnog broja studenata.

Grafikon 98: Broj studenata koji su izabrali izborni predmet.



6.3. Diskusija

Ukoliko se detaljnije pogledaju rezultati anketiranja, može se primetiti da je simulator na prvom ispitivanju (na početku semestra) dobio nešto nižu ocenu nego na drugom (na kraju semestra).

Niža ocena na početku se može objasniti time što su studenti upotrebu simulatora na predmetu videli samo kao dodatnu obavezu, da bi tokom semestra uvideli na koji im način pomaže da razumeju oblast kojom se bave. Takođe, na početku studenti nisu bili upoznati sa sadržajem predmeta pa im je zbog toga simulator možda delovao suviše komplikovano. S obzirom da 2007., 2008. i 2009. godine sistem nije bio u upotrebi, poređenjem podataka na grafikonima može se doći do zaključka da studentsko znanje nije variralo tokom posmatranog perioda.

Ocene u prva dva ispitna roka (junski i julski) za predmet Inteligentni informacioni sistemi, kao i druge predmete koje je slušala ista grupa studenata, pokazuju da je više studenata položilo Inteligentne informacione sisteme nego druge predmete. U naredna dva ispitna roka (septembar i oktobar), predmet je bio među onima sa najmanje studenata koji su se prijavili za polaganje. To je pokazalo da su studenti koji su aktivno učestvovali u laboratorijskim vežbama i koristili simulator lakše polagali ispit u prvim ispitnim rokovima.

Poređenje između eksperimentalne i kontrolne grupe tokom posmatranog perioda je pokazalo da je eksperimentalna grupa postigla za 0,79 veću ocenu nego kontrolna grupa.

Za povećanje broja studenata koji slušaju Inteligentne informacione sisteme sa 30% na 51% su delimično zaslužni i studenti čije osnovne interesovanje nisu informacione tehnologije. To je samo još jedan pokazatelj da su složeni koncepti u oblasti inteligentnih sistema postali atraktivniji i lakši za razumevanje zahvaljujući korišćenju vizuelnih i interaktivnih alata kao što je predstavljeni simulator.

7. ZAKLJUČAK

Vizuelne softverske simulacije su se u nastavi pokazale kao odlično rešenje za prezentovanje određenih tehnika rada, ili algoritama u slučaju inteligentnih sistema. Na taj način se, uz obavezno poštovanje određenih metodičkih pravila, mogu postići izuzetno dobri rezultati. Zahvaljujući razvoju softvera koji poštuje ta pravila, može se dobiti edukativni alat koji može znatno poboljšati kvalitet nastave.

Korišćenjem softverskih simulacija moguće je koristiti modele situacija iz realnog sveta uz strogo definisane uloge svih učesnika u okruženju bogatom parametrima koji pomažu u donošenju poslovnih odluka. Na ovaj način je moguće steći praktičan uvid u način funkcionisanja poslovnih procesa u modernom preduzeću. Postoji puno simulatora ovakve vrste, a u radu je dat pregled najpoznatijih. U svakom od prikazanih simulatora postoji samo mogućnost dobijanja detaljnih izveštaja i bilansa na osnovu kojih je potrebno donositi odluke, ali ne postoji bilo kakva mogućnost da korisnici detaljnije uđu u ovaj proces i dobiju pomoć u analiziranju mnoštva raznih parametara.

U ovom segmentu kao pogodno rešenje nametnuli su se inteligentni sistemi, a u okviru njih posebno algoritmi mašinskog učenja i algoritmi pretraživanja. Ovi algoritmi, kao i načini pomoću kojih se može olakšati njihovo učenje, čine osnovu ovog rada. Baš zbog toga, kao prirodno rešenje ukazalo se razvijanje vizuelnog softverskog simulatora koji bi se bavio algoritmima mašinskog učenja i pretraživanja.

Nakon proučavanja preporuka vodećih strukovnih organizacija i vodećih svetskih univerziteta, odlučeno je da se u okviru simulatora obrade sledeći algoritmi mašinskog učenja: stabla odlučivanja, klasterovanje, Naive Bayes i perceptroni. Između algoritama pretraživanja, odlučeno je da se koriste pretraživanje po širini i dubini, planinarenje i A*. Kako bi bilo moguće pristupiti izradi softvera, prvo je postavljena teorijska osnova rada praćena pregledom literature. Navedena su trenutna dostignuća u ovim oblastima, uz objašnjenja načina rada navedenih algoritama i predstavljanje njihovih karakteristika.

Razvijeni simulator omogućava korisnicima da vide kako izabrani algoritmi funkcionišu. Kompletan sistem je zasnovan na istom principu rada - moguće je zadati ulazni skup podataka i na njega primeniti željeni algoritam, a potom, korak po korak, pratiti na koji način se izvršava. Tokom same simulacije, prikazuju se i detaljne informacije o tome u kojoj se trenutno fazi nalazi algoritam kako bi se lakše razumeo način njegovog rada.

U radu je objašnjen i način rada svakog od algoritama, a navedeni su i detalji njihove tehničke realizacije. Naveden je model svakog algoritma, kao i opis kako je implementiran u okviru odgovarajućeg modula.

Posebno poglavlje je posvećeno detaljima tehničke realizacije predstavljenog sistema za vizuelnu simulaciju algoritama inteligentnih sistema, kao i rešenja problema koji su se pojavili tokom razvoja. Prikazano je i kako je ceo sistem organizovan po modulima na taj način da svaki obrađeni algoritam predstavlja zasebnu celinu. Izuzetak su algoritmi pretraživanja koji zbog svoje strukturne sličnosti čine zajednički modul, a u okviru koga je moguće izabrati algoritam koji će biti simuliran.

Za razvoj softverskog sistema odabrana je Java platforma. Na ovaj način, softver može da se izvršava na velikom broju različitih sistema, a da se pritom zadrži isti izgled i osećaj prilikom korišćenja. Naravno, ne postoje ni troškovi licenciranja jer se radi o platformi otvorenog izvornog koda. U razvoju su korišćene i dodatne biblioteke. Za iscrtavanje grafičkih elemenata, počevši od najjednostavnijih oblika do složenijih struktura, korišćena je Java AWT tehnologija. Komponente korisničkog interfejsa i njihov raspored su napravljeni pomoću biblioteke Java Swing. Za rad sa stablima i grafovima korišćena je biblioteka JUNG, pomoću koje je olakšan proces njihovog prikazivanja.

Na samom kraju rada, predstavljeni su rezultati istraživanja koje je uključilo iskustva sakupljena korišćenjem predstavljenog simulatora na predmetu Inteligentni informacioni sistemi na Poslovnom fakultetu Valjevo Univerziteta Singidunum. Simulator se, počevši od prve razvijene verzije, na ovom predmetu koristi u nastavi četiri godine za samostalno učenje, za laboratorijske vežbe, i kao alat za nastavnike koji omogućava da studentima brže prenose znanje i da ih testiraju.

Predstavljeni su numerički pokazatelji koji dokazuju prednosti korišćenja opisanog simulatora. Broj studenata koji je izabrao izborni predmet Inteligentni informacijski sistemi je porastao sa 30% na 51% od početka korišćenja softvera u nastavi. Tokom tog perioda, procenat studenata koji su položili predmet je porastao sa 68% na 88%, dok je prosečna ocena povećana sa 7,45 na 8,39.

Ukoliko bi se posmatrali načini na koje bi sistem mogao da se poboljša, došlo bi se do moguće nadogradnje sistema u smislu proširenja njegovih funkcionalnosti. Na primer, moguće bi bilo obraditi i neke od novijih varijanti prikazanih algoritama. Ako bi se ukazala potreba, mogao bi se dodati i kompletan modul i obraditi još neki od algoritama inteligentnih sistema koji nije ušao u izbor za ovu verziju simulatora. Moguće bi bilo i orijentisati se ka razvoju modula za vrednovanje aktivnosti pri radu u simulatoru. Na taj način, sistem bi mogao da se pored demonstriranja rada algoritama koristi i za ocenjivanje. Takav pristup bi sigurno mogao da predstavlja veliku dodatnu prednost pri korišćenju ovakvog softvera.

LITERATURA

- [1.] Wolfe J., A history of business teaching games in English-speaking and post-socialist countries, *Simulation & Gaming* 24, 1993, str. 446-467.
- [2.] Faria J. A., Nulsen R., *Developments in Business Simulation & Experiental Exercieses*, Volume 23, University of Windsor, Xavier University, 1996.
- [3.] Meier A.C., Newell W.T., Pazer H.L., *Simulation in Business Economics*, Prentice-Hall Publishing Company, Englewood Cliffs, N.J., 1969.
- [4.] Andlinger G.R., Business games—play one!, *Harvard Business Review* 36 (11), 1958, str. 277–284.
- [5.] Faria. J. A, *Business Simulation Games After Thirty Years*, Guide to Business Gaming and Experiental Learning, Nichols Pub Co, 1990.
- [6.] Farrell C., Perceived effectiveness of simulations in international business pedagogy: an exploratory analysis, *Journal of Teaching in International Business* 16 (3), 2005, str. 71–88.
- [7.] Musselwhite C., University Executive Education Gets Real, *Training and Development Magazine* 6, 2006, str. 57–59.
- [8.] Gibson D., Aldrich C., Prensky M., *Games and Simulations in Online Learning*, Information Science Publishing, Hershey, 2007.
- [9.] Kibbee J.M., Craft C.J., Nanus B., *Management Games*, Reinhold Publishing Company, New York, 1961.
- [10.] Summers G.J., Today's business simulation industry, *Simulation & Gaming* 35, 2004, str. 208–241.
- [11.] Gredler E. M., *Games and Simulations and their Relationships to Learning*, Handbook of Research on Educational Communications, Lawrence Erlbaum Associates Publishers, Mahwah, NJ, 2004.

- [12.] Dittrich J. E., Realism in Business Games: A Three Game Comparison, University of Kentucky, Lexington, Kentucky, 1977.
- [13.] Wawer M., Milosz M., Muryjas P., Rzemieniak M., Business Simulation Games in Forming of Students' Entrepreneurship, International Journal of Euro-Mediterranean Studies, Volume 3, EMUNI University i University of Nova Gorica, Portorož, 2010.
- [14.] ACM, IEEE, Computer Science Curriculum 2008: An Interim Revision of CS 2001, str. 79-85, dostupno na: <http://www.acm.org/education/curricula/ComputerScience2008.pdf>, datum pristupa: 21.4.2014.
- [15.] G. Stahl, T. Koschmann, D. Suthers, Computer-supported collaborative learning: An historical perspective, dostupno na: http://www.gerrystahl.net/cscl/cscl_english.htm, datum pristupa: 17.5.2014.
- [16.] Jovanovic, N., Popovic, R., Markovic, S. and Jovanovic, Z., Web laboratory for computer network, Computer Applications in Engineering Education, Wiley, 2012, str. 493-502.
- [17.] http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Naive_Bayes_classifier.html, datum pristupa: 22.5.2014.
- [18.] Grbanovic, N., Nikolic, B. and Djordjevic, J., The VSDS environment based laboratory in computer architecture and organisation, Computer Applications in Engineering Education, Wiley, 2011, str. 685-696.
- [19.] Kayisli, K., Tuncer, S. and Poyraz, M., An educational tool for fundamental DC-DC converter circuits and active power factor correction applications. Computer Applications in Engineering Education, Wiley, 2013, str. 113-134.
- [20.] ACM, IEEE, Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering, dostupno na: http://www.acm.org/education/education/curric_vols/CE-Final-Report.pdf, datum pristupa: 23.4.2014.

- [21.] Rubinstein F. M., Firstenberg R. I., Patterns of Problem Solving, Prentice Hall, Englewood Cliffs, New Jersey, 1995.
- [22.] Čupić M., Suknović M., Odlučivanje, Fakultet organizacionih nauka, Beograd, 2010.
- [23.] Turban E., Volonino L., Information Technology for Management: Improving Strategic and Operational Performance, Wiley, 2011.
- [24.] Pražić J., Bukvić B., Torbica V., Šćepanović I., Marketing informacioni sistemi, Fakultet za menadžment, Novi Sad, 2010.
- [25.] Druzdel J. M., Flynn R. R., Decision Support Systems, University of Pittsburgh, Pittsburgh, 2002.
- [26.] Holsapple W. C., DSS Architecture and Types, Handbook on Decision Support Systems 1, Springer-Verlag, Heidelberg, 2008.
- [27.] ACM, IEEE, Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering, dostupno na: http://www.acm.org/education/education/curric_vols/CE-Final-Report.pdf, datum pristupa: 23.4.2014.
- [28.] ACM, IEEE, Computer Science Curriculum 2008: An Interim Revision of CS 2001, December 2008.
- [29.] ACM, IEEE, Computer Science Curricula 2013, Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, December 2013.
- [30.] Wu X., Kumar V., Quinlan R., Ghosh J., Yang Q., Motoda H., McLachlan G., Ng A., Liu B., Yu P., Zhou Z., Steinbach M., Hand D., Steinber D., Top 10 algorithms in data mining, Springer-Verlag, London, 2007.
- [31.] <http://www.cs.berkeley.edu/~russell/classes/cs194/f11/syllabus.html>, datum pristupa: 8.5.2014.
- [32.] <http://cs229.stanford.edu/schedule.html>, datum pristupa: 9.5.2014.

- [33.] <http://online.stanford.edu/course/algorithms-design-and-analysis-part-1>, datum pristupa: 13.6.2014.
- [34.] <http://online.stanford.edu/course/algorithms-design-and-analysis-part-2>, datum pristupa: 13.6.2014.
- [35.] <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/syllabus/>, datum pristupa: 9.5.2014.
- [36.] <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-spring-2005/syllabus/>, datum pristupa: 9.5.2014.
- [37.] Design and Analysis of Algorithms , MIT, <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-design-and-analysis-of-algorithms-spring-2012/Syllabus/course-objectives-and-outcomes/>, datum pristupa: 31.4.2014.
- [38.] <http://www.cs.cmu.edu/~epxing/Class/10701-08s/lecture.html>, datum pristupa: 11.5.2014.
- [39.] <http://www.maastrichtuniversity.nl/web/Main/Sitewide/courseDetailpagina/IntelligentSearchGames507310362013EnKEN4123.htm> datum pristupa: 13.6.2014.
- [40.] <http://www.maastrichtuniversity.nl/web/Main/Sitewide/courseDetailpagina/DataMining505852302013EnKEN4113.htm>, datum pristupa: 13.6.2014.
- [41.] <http://ri4es.etf.rs/index.html>, datum pristupa: 5.3.2014.
- [42.] W. Peng, J. Chen and H. Zhou, An Implementation of ID3 - Decision Tree Learning Algorithm, University of new South Wales, School of Computer Science and Engineering, Sydney, Australia
- [43.] <http://web.arch.usyd.edu.au/~wpeng/DecisionTree2.pdf>, datum pristupa: 11.5.2014.
- [44.] AI Decision Tree Learner Applet, dostupno na: <http://www.doc.ic.ac.uk/~tora/previous/AIDTL/DecisionTree.html>, datum pristupa: 11.5.2014.

- [45.] A. Tristan and S. Eisenbach, CLASE: Cursor Library for A Structured Editor (Tool Demo), Haskell '08: Proceedings of the first ACM SIGPLAN symposium on Haskell, 2008.
- [46.] J. Spehr, S. Winkelbach, The k-means algorithm, dostupno na: <http://www.rob.cs.tu-bs.de/content/04-teaching/06-interactive/Kmeans/Kmeans.html>), datum pristupa: 15.5.2014.
- [47.] Clustering Applet, dostupno na: <http://webdocs.cs.ualberta.ca/~yaling/Cluster/Applet/Code/Cluster.html>, datum pristupa: 5.5.2014.
- [48.] Y. Pei and O. Zaïane, A synthetic data generator for clustering and outlier analysis, Technical Report TR06-15, Department of Computing Science, University of Alberta, June 2006.
- [49.] Classification Applet, dostupno na: <http://www.cs.technion.ac.il/~rani/LocBoost/>, datum pristupa: 5.5.2014.
- [50.] Y. Baram and R. El-Yaniv and K. Luz, Online Choice of Active Learning Algorithms, Journal of Machine Learning Research, March 2004, str. 255-291
- [51.] Neurocomputers - Perceptron training algorithm, dostupno na: <http://www.eee.metu.edu.tr/~halici/courses/543java/NNOC/Perceptron.html>, datum pristupa: 7.5.2014.
- [52.] Halici U., Atalay V., Leblebicioglu K., Arkan A., Mulayim M., Ceran S., Distance Education on Internet: A Java Based Image Processing Course in METU., Proc. of 2nd International Symposium on Distance Education, Ankara, Turkey, May 1998, str. 428-437
- [53.] Borm I., The Design and Implementation of a Multi-Agent Soccer Simulator as a Tool for AI Research and Education, Delft University of Technology, Delft, 2006, str. 3.
- [54.] WEKA, dostupno na: <http://www.cs.waikato.ac.nz/ml/weka/>, datum pristupa: 11.5.2014.

- [55.] Hall M., Frank E., Holmes G., Pfahringer G., Reutemann P., Witten I., The WEKA Data Mining Software: An Update, ACM SIGKDD Explorations Newsletter, Volume 11 Issue 1, New York, NY, USA, 2009.
- [56.] Witten I., Frank E., Hall M., Data Mining - Practical Machine Learning Tools and Techniques, 3rd edition, Morgan Kaufmann, 2011.
- [57.] F. Borrajo, Y. Bueno, I. de Pablo, B. Santos, F. Fernández, J. García, I. Sagredo, SIMBA: A simulator for business education and research, Decision Support Systems, Volume 48, Issue 3, February 2010, str. 498–506
- [58.] Borrajo F., Bueno Y., Fernández F., García J., de Pablo I., Sagredo I., Santos B., Business Simulators for Business Education and Research: SIMBA Experience, Computer Games as Educational and Management Tools: Uses and Approaches, Hershey, 2011, str. 229-246
- [59.] MIT Open CourseWare, dostupno na: <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/demonstrations/>, datum pristupa: 29.3.2014.
- [60.] AISpace, dostupno na: <http://www.aispace.org/>, datum pristupa: 30.3.2014.
- [61.] Knoll B., Kisynski J., Carenini G., Conati C., Mackworth A., Poole D. L., AIspace: Interactive Tools for Learning Artificial Intelligence, Proceedings of the AAAI AI Education Colloquium, Chicago, IL, 2008.
- [62.] AIMA3e, dostupno na: <https://code.google.com/p/aima-java/>, datum pristupa: 30.3.2014.
- [63.] AI-Search, dostupno na: <http://www.cs.rmit.edu.au/AI-Search/Product/>, datum pristupa: 30.3.2014.
- [64.] Data Structure Visualisations, dostupno na: <https://www.cs.usfca.edu/~galles/visualization/>, datum pristupa: 30.3.2014.
- [65.] Quinlan R., Machine Learning, Kluwer Academic Publishers, Boston, 1986.
- [66.] Chen J.; Luo D.; Mu F., An improved ID3 decision tree algorithm, Computer Science & Education, ICCSE '09, 4th International Conference, 2009, str. 127.

- [67.] Elyassami S., Idri A., Applying Fuzzy ID3 Decision Tree for Software Effort Estimation, IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, 2011, str. 131-138.
- [68.] Suknovic M., Delibasic B., Jovanovic M., Vukicevic M., Becejski-Vujaklija D., Obradovic Z., Reusable components in decision tree induction algorithms, Computational Statistics, Volume 27, Issue 1, March 2012, str. 127-148.
- [69.] Nikolić B., Ekspertski sistemi, Elektrotehnički fakultet, Beograd, 2011.
- [70.] Russell S., Norvig P., Veštačka inteligencija – savremeni pristup, CET, Beograd, 2011, str. 764-765.
- [71.] Shannon C. E., A Mathematical Theory of Communication, The Bell System Technical Journal, Vol. 27, 1948, str. 379-423.
- [72.] Mitchell T., Machine Learning, McGraw Hill, Boston, 1997, str. 57.
- [73.] Mathur N., Kumar S., Kumar S., Jindal R., The Base Strategy for ID3 Algorithm of Data Mining Using Havrda and Charvat Entropy Based on Decision Tree, International Journal of Information and Electronics Engineering, vol. 2, no. 2, 2012, str. 253-258.
- [74.] Jyothirmayi T., Reddy S., An Algorithm for Better Decision Tree, International Journal on Computer Science and Engineering (IJCSE), 2(9), 2010, str. 2827-2830.
- [75.] Alxploratorium, University of Alberta, dostupno na:
<http://webdocs.cs.ualberta.ca/~aixplore/learning/DecisionTrees/InterArticle/3-DecisionTree.html>, datum pristupa: 14.6.2014.
- [76.] Pang-Ning T., Steinbach M., Kumar V., Introduction to Data Mining, Addison-Wesley, Boston, 2006, str. 487.
- [77.] Lloyd S. P., Least Squares Quantization in PCM, IEEE Trans. Information Theory, vol. 8, 1982, str. 129-137.
- [78.] Faber V., Clustering and the Continuous k-means Algorithm, Los Alamos Science, vol. 2, 1994, str. 138-144.

- [79.] Forgey E., Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classification, *Biometrics*, vol. 21, 1965.
- [80.] J. MacQueen, Some Methods for Classification and Analysis of Multivariate observations, *Proceedings Fifth Berkeley Symp. Math. Statistics and Probability*, vol. 1, 1967, str. 281-296.
- [81.] Kanungo T., Mount D.M., Netanyahu N.S., Piatko C.D., Silverman R., Wu A.Y., An efficient k-means clustering algorithm: analysis and implementation, *Pattern Analysis and Machine Intelligence, IEEE Transactions*, vol.24, no.7, July 2002. str. 881-892.
- [82.] Tan Pang-Ning, Steinbach M., Kumar V., op. cit., str. 497-450.
- [83.] Cheeseman P., Stutz J., Bayesian classification (AutoClass): Theory and results. In *Advances in knowledge discovery and data mining*, AAAI Press, Menlo Park, CA, 1996, str. 153–180.
- [84.] Domingos P., Elkan M., On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29, 1997, str. 103–130.
- [85.] Lowd D., Domingos P., Naive Bayes Models for Probability Estimation, *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.
- [86.] Ting S.L., Ip W.H., Tsang A., Is Naive Bayes a Good Classifier for Document Classification?, *International Journal of Software Engineering and Its Applications*, Vol. 5, No. 3, July, 2011, str. 37-46.
- [87.] Flach P. A., Lachiche N., Naive Bayesian Classification of Structured Data, *Machine Learning*, Volume 57, Issue 3, December 2004, str. 233-269.
- [88.] Chakrabarti S., Roy S., Soundalgekar M.V., Fast and accurate text classification via multiple linear discriminant projection, *The VLDB Journal The International Journal on Very Large Data Bases*, 2003, str. 170–185.
- [89.] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, In *Proceedings: Machine Learning: ECML - 98*, 10th European Conference on Machine Learning, 1998, str. 137–142.

- [90.] McCann S., Lowe D. G., Local Naive Bayes Nearest Neighbor for Image Classification, CoRR, 2011.
- [91.] McCulloch W.S., Pitts W., A Logical Calculus of Ideas Immanent in Nervous Activity, Bull. Mathematical Biophysics, Vol. 5, 1943, str. 115-133.
- [92.] Nguyen, D.H.; Widrow, B., Neural networks for self-learning control systems, Control Systems Magazine, IEEE , vol.10, no.3, April 1990, str. 18-23.
- [93.] Jain A.K., Mao J., Mohiuddin K. M., Artificial neural networks: a tutorial, Computer , vol.29, no.3, Mar 1996, str. 31-44.
- [94.] Picton P., Neural Networks, Palgrave, Hampshire, 2000, str. 29.
- [95.] Gallant S. I., Perceptron-Based Learning Algorithms, IEEE Transactions on Neural Networks, Vol. 1, No. 2, June 1990, str. 179-191.
- [96.] Dasgupta S., Kalai A. T., Monteleoni C., Analysis of Perceptron-Based Active Learning, Journal of Machine Learning Research, 10, 2009, str. 281-299.
- [97.] Tomašević M., Struktura podataka, Akademska misao, Beograd, 2005., str. 209.
- [98.] Marinescu R., Kask K., Dechter R., Systematic vs. non-systematic algorithms for solving the MPE task, *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence (UAI'03)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002, str. 394-402.
- [99.] Ajwani D., Dementiev R., Meyer., A Computational Study of External-Memory BFS Algorithms, Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, Florida, USA, 2006.
- [100.] Kurant M., Markopoulou A., Thiran P., On the bias of BFS (Breadth First Search), International Teletraffic Congress (ITC 22), 2010., str. 2.
- [101.] Breadth-First Search Traversal Algorithm, Kent University, dostupno na: <http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/breadthSearch.htm>, datum pristupa: 2.3.2014.

- [102.] Beamer S., Asanović K., Patterson D., Direction-optimizing breadth-first search, *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12)*, Article 12, IEEE Computer Society Press, Los Alamitos, CA, USA, 2012.
- [103.] <https://www.cs.bu.edu/teaching/c/tree/breadth-first/>, datum pristupa: 2.3.2014.
- [104.] Alsuwaiyel M. H., Algorithms Design Techniques and Analysis, World Scientific Publishing, Beijing, 2003, str. 269.
- [105.] Živković D., Osnove dizajna i analize algoritama, Računarski fakultet, CET, Beograd, 2007., str. 263
- [106.] Tarjan R., Depth first search and linear graph algorithms, SIAM Journal on Computing, USA, 1972.
- [107.] King D. J., Launchbury J., Lazy Depth-First Search and Linear Graph Algorithms in Haskell In GLA (93), str. 145-155
- [108.] Dow P. A., Korf R. E., Duplicate avoidance in depth-first search with applications to tree width, *Proceedings of the 21st international joint conference on Artificial intelligence (IJCAI'09)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009., str. 480-485.
- [109.] Deng X., Yao Y., Jiarui C., Improving Depth-First Search Algorithm of VLSI Wire Routing with Pruning and Iterative Deepening, *Emerging Research in Artificial Intelligence and Computational Intelligence, Communications in Computer and Information Science*, Volume 237, Springer Berlin Heidelberg, 2011, str. 100-107
- [110.] Selman B., Gomes C. P., Hill-climbing Search, *Encyclopedia of Cognitive Science*, John Wiley & Sons, 2006.
- [111.] Langley P., Gennari J. H., Iba W., Hill-Climbing Theories of Learning, Interim rept., California University Irvine School of Information and Computer Science, 1987.

- [112.] Papadias D., Hill Climbing Algorithms for Content-Based Retrieval of Similar Configurations, Proceedings of the ACM Conference on Information Retrieval (SIGIR), Athens, July 24-28, 2000.
- [113.] Smitley D. L., Lee I., Comparative Analysis of Hill Climbing Mapping Algorithms, Technical Reports (CIS), Department of Computer & Information Science, University of Pennsylvania, USA, 1988.
- [114.] Wilt C., Thayer J., Ruml W., A Comparison of Greedy Search Algorithms, Proceedings of the Third Annual Symposium on Combinatorial Search (SOCS-10), Atlanta, GA (USA), 2010.
- [115.] Indiana University, Artificial Intelligence Course, dostupno na: <http://homepages.ius.edu/RWsISMAN/C463/>, datum pristupa: 30.3.2014.
- [116.] Sosic R., Gu J., 3.000.000 Queens in less than one minute. *SIGART Bull.* 2, 1991, str. 22-24.
- [117.] Jacobson S. H., Yücesan E., Analyzing the Performance of Generalized Hill Climbing Algorithms, Journal of Heuristics, Volume 10, Issue 4, Kluwer Academic Publishers, 2004, str. 387-405
- [118.] Luby M., Sinclair A., Zuckerman D., Optimal speedup of Las Vegas algorithms, Information Processing Letters 47, 1993, str. 173-180.
- [119.] Hart P. E., Nilsson N. J., Raphael B., A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics, 1968.
- [120.] Nosrati M., Karimi R., Hasanvand H. A., Investigation of the * (Star) Search Algorithms: Characteristics, Methods and Approaches, World Applied Programming, Vol (2), No (4), 2012, str. 251-256.
- [121.] Cui X., Shi H., A*-based Pathfinding in Modern Computer Games, IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, January 2011, str. 125-130.

- [122.] Zhang Z., Sturtevant N. R., Holte R., Schaeffer J., Felner A., A* search with inconsistent heuristics, Proceedings of the 21st international joint conference on Artificial intelligence (IJCAI'09), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009, str. 634-639.
- [123.] Grant S., Williams D., Using the A-Star Path-Finding Algorithm for Solving General and Constrained Inverse Kinematics Problems, Orion Transfer, 2008.
- [124.] Martelli A., On the Complexity of Admissible Search Algorithms, Artificial Intelligence, 8(1):1-13, 1977.
- [125.] Horstmann C., Cornell G., Java 2 – Osnove, CET, Beograd, 2007.
- [126.] <http://www.javaworld.com/javaworld/jw-07-1996/jw-07-awt.html>, датум приступа: 21.4.2014.
- [127.] <http://edn.embarcadero.com/article/26970>, датум приступа: 20.1.2014.
- [128.] <http://docs.oracle.com/javase/tutorial/ui/overview/intro.html>, датум приступа: 22.4.2014.
- [129.] <http://jung.sourceforge.net/>, датум приступа: 22.4.2014.
- [130.] O'Madadhain J., Fisher D., Smyth P., White S., Boey Y-B., Analysis and Visualization of Network Data using JUNG, Journal of Statistical Software, Volume VV, Issue II, 2005.
- [131.] Roach C., Building Decision Trees in Python, O'Reilly Media, dostupno na: http://oreilly.com/pub/a/python/2006/02/09/ai_decision_trees.html?page=1/, datum pristupa: 14.6.2014.
- [132.] Fausett L., Fundamentals of Neural Networks - Architectures Algorithms and Applications, Prentice Hall, Upper Saddle River, 1994