RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts

2.praktiskais darbs
mācību priekšmetā
“Mākslīgais intelekts”

Mašīnmācīšanās algoritmu lietojums

Saite uz darbu: https://github.com/runcis/MI_PD2_171RDB359

Izstrādāja: Rinalds Pikše

171rdb359

Pārbaudīja: Alla Anohina Naumeca

2022./23. māc. Gads

Saturs

Uzdevums

I daļa - Datu pirmapstrāde/izpēte

1. Ir jāizvēlas un jāapraksta datu kopa, pamatojoties uz informāciju, kas sniegta krātuvē, kurā datu kopa ir pieejama.
2. Ja no krātuves iegūtā datu kopa nav formātā, ar kuru ir viegli strādāt (piemēram, komatatdalītās vērtības vai .csv fails), ir jāveic tās transformācija vajadzīgajā formātā. Datu kopas failam ir jābūt n × d tabulai, kur d ir datu pazīmju (atribūtu) skaits un n ir datu objektu skaits. Tabulas kolonas ir jāsakārto šādā secībā: datu objekta ID, datu objekta klases iezīme un pēc tam visu pazīmju (atribūtu) vērtības.
3. Ja kādu pazīmju (atribūtu) vērtības ir tekstveida vērtības (piemēram, yes/no, positive/neutral/negative, u.c.), tās ir jātransformē skaitliskās vērtībās.
4. Ja kādiem datu objektiem trūkst atsevišķu pazīmju (atribūtu) vērtības, ir jāatrod veids, kā tās iegūt, studējot papildu informācijas avotus.
5. Ir jāatspoguļo datu kopa vizuāli un jāaprēķina statistiskie rādītāji:
    a. ir jāizveido vismaz divas 2- vai 3-dimensiju izkliedes diagrammas(scatter plot), kas ilustrē klases atdalāmību, balstoties uz dažādām pazīmēm (atribūtiem); studentam ir jāizvairās izmantot datu objekta ID kā mainīgo izkliedes diagrammā;
    b. ir jāizveido vismaz 2 histogrammas, kas parāda klašu atdalīšanu, pamatojoties uz interesējošām pazīmēm (atribūtiem);
    c. ir jāatspoguļo 2 interesējošo pazīmju (atribūtu) sadalījums;
    d. ir jāaprēķina statistiskie rādītāji (vismaz vidējās vērtības un dispersiju).

II daļa – Nepārraudzītā mašīnmācīšanās

1. Jāpielieto divi studiju kursā apskatītie nepārraudzītās mašīnmācīšanās algoritmi: (1) hierarhiskā klasterizācija un (2) K-vidējo algoritms.
2. Hierarhiskās klasterizācijas algoritmam ir jāveic vismaz 3 eksperimenti, brīvi mainot hiperparametru vērtības, un analizējot algoritma darbību;
3. K-vidējo algoritmam ir jāveic eksperimenti ar vismaz 5 k vērtībām, jāaprēķina Silhouette Score, un jāanalizē algoritma darbība

III daļa – Pārraudzītā mašīnmācīšanās

1. Ir jāizvēlas vismaz divi pārraudzītās mašīnmācīšanās algoritmi, kas ir paredzēti klasifikācijas uzdevumam. Studenti drīkst izmantot studiju kursā aplūkotos algoritmus vai arī jebkurus citus algoritmus, ko piedāvā Orange rīks klasifikācijas uzdevumam.
2. Ir jāsadala datu kopa apmācību un testa datu kopās.
3. Katram algoritmam, lietojot apmācību datu kopu, ir jāveic vismaz 3 eksperimenti, mainot algoritma hiperparametru vērtības un analizējot algoritmu veiktspējas metrikas;
4. Katram algoritmam ir jāizvēlas tas apmācītais modelis, kas nodrošina labāko algoritma veiktspēju;
5. Katra algoritma apmācītais modelis ir jāpielieto testa dat u kopai.
6. Ir jānovērtē un jāsalīdzina apmācīto modeļu veiktspēja.

I daļa – Datu pirmapstrāde/izpēte

Izvēlētā datu kopa

Kā datu kopu izvēlējos kopu “Stellar Classification Dataset - SDSS17” un tā ir pieejama šeit https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17. Šīs datu kopas autors ir Federico Soriano Palacios. Avots datiem ir SDSS(Sloan Digital Sky Survey) optiskais teleskops kas atrodas ASV, Ņūmeksikas štatā, ‘Apache Point’ observatorijā. Datu kopa satur ierakstus par observatorijā novērotajām zvaigznēm, galaktikām un kvazāriem. Katram ierakstam ir 18 parametri - 17 īpašību parametri un 1 klasifikatora parametrs, kas izsaka, vai ieraksts ir zvaigzne, galaktika vai kvazārs.

Observējot debesis, ir iespējams saskatīt daudz gaismas avotu – no cilvēka skatupunkta tās visas šķiet ka ir zvaigznes, bet tā nav. Es pats, izmantojot mazu teleskopu esmu novērojis, ka ir iespējams redzēt mūsu saules sistēmā esošās planētas – Marsu, Jupiteru, Saturnu ar neapbruņotu aci. Bet kad tās apskata bez teleskopa tās izskatījās pēc zvaigznēm. Šo problēmu, tikai zinātniskā līmenī – nespēju atšķirt gaismas avotus, mēģina risināt šo datu autori. Kad Observatorijas optisko teleskopu notēmē uz visumu, tas reģistrē simtiem tūkstošu gaismas avotu, katram no tiem ir savi ultravioletās, infrasarkanās gaismas filtru lasījumi, sava atrašanās vieta un citi parametri. Astronomi ir identificējuši daudz gaismas avotu mūsu debesīs, tāpēc šiem ievāktajiem debesu lasījumiem ir piesaistītas klasifikācijas, kas norāda, vai objekts ir zvaigzne, galaktika vai kvazārs(spīdošs galaktikas centrs). Balstoties uz šiem datiem un klasifikāciju, mēs varam izmantot šo datu setu lai atrastu pazīmes, kas ļaus ātrāk klasificēt jaunatklātus debesu objektus. Šī darba ietvaros arī analizēšu, kāda ir parametru ietekme uz datu iedalīšanu klasēs.

Datu kopas saturs

(Iegūts no Stellar Classification Dataset - SDSS17
https://www.kaggle.com/datasets/fedesoriano/stellar-classification-dataset-sdss17)

1. obj_ID = Objekta identifikators.
2. alpha = Rektasencijas lenķis (pēc 2000. gada diskretizācijas perioda)
3. delta = Deklinācijas leņķis (pēc 2000. gada diskretizācijas perioda)
4. u = Ultravioletais filtrs, fotometriskajā sistēmā
5. g = Zaļais filtrs, fotometriskajā sistēmā
6. r = Sarkanais filtrs, fotometriskajā sistēmā
7. i = Tuvās infrasarkanās gaismas filts, fotometriskajā sistēmā
8. z = Infrasarkanās gaismas filts, fotometriskajā sistēmā
9. run_ID = Specifiskā skenējuma id
10. rereun_ID = Atkārtotā skenējuma id, lai specificētu, kā bilde bija procesēta
11. cam_col = Kameras kolonna, lai identificētu skanējuma līniju
12. field_ID = Lauka id
13. spec_obj_ID = Unikāls id, izmantots priekš optiskiem spekroskopiskiem objetkiem.
14. class = Objekta klase (zvaigzne, galaktika vai kvazārs)
15. redshift = Sarkanā nobīde, balstīta uz viļņa garuma pieaugumu
16. plate = Plates Id,identificē teleskopa plati
17. MJD = Datums, kurā novērojums tika veikts.

18. fiber_ID = šķiedra, ar kuru tikai veikts novērojums teleskopā.

Datu kopas modifikācijas

Apskatot sarakstu ar atribūtiem, varam redzēt, ka daudzi no atribūtiem nav unikāli priekš gaismas objektiem un attiecas uz teleskopa un laika vienībām, kurās datu ieraksti ir saņemti. Šī iemesla dēļ, es izvēlos noņemt no datu kopas sekojošās kolonnas: run_ID, rerun_ID, cam_col, field_ID, spec_obj_ID, plate, MJD, fiber_ID. Izņemot šīs kolonnas, mums paliek:

obj_ID = Objektu identificējošs kods.
alpha = Rektasencijas lenķis (pēc 2000. gada diskretizācijas perioda) [grādi°]
delta = Deklinācijas leņķis (pēc 2000. gada diskretizācijas perioda) [grādi°]
u = Ultravioletās gaismas filtrs, fotometriskajā sistēmā [nm]
g = Zaļās gaismas filtrs, fotometriskajā sistēmā [nm]
r = Sarkanais gaismas filtrs, fotometriskajā sistēmā [nm]
i = Tuvās infrasarkanās gaismas filts, fotometriskajā sistēmā [nm]
z = Infrasarkanās gaismas filtrs, fotometriskajā sistēmā [nm]
class = Objekta klase (zvaigzne, galaktika vai kvazārs)
redshift = Sarkanā nobīde, balstīta uz vilņa garuma pieaugumu

Dati ir saglabāti csv faila formātā un tikai viena kolonna nesatur skaitliskus datus – klasifikācijas kolonna. Lai varētu strādāt ar šiem datiem Orange rīkā, pārveidošu šo kolonnu skaitliskās vērtībās: 1 – Zvaigzne; 2 – Galaktika; 3 – Kvazārs.

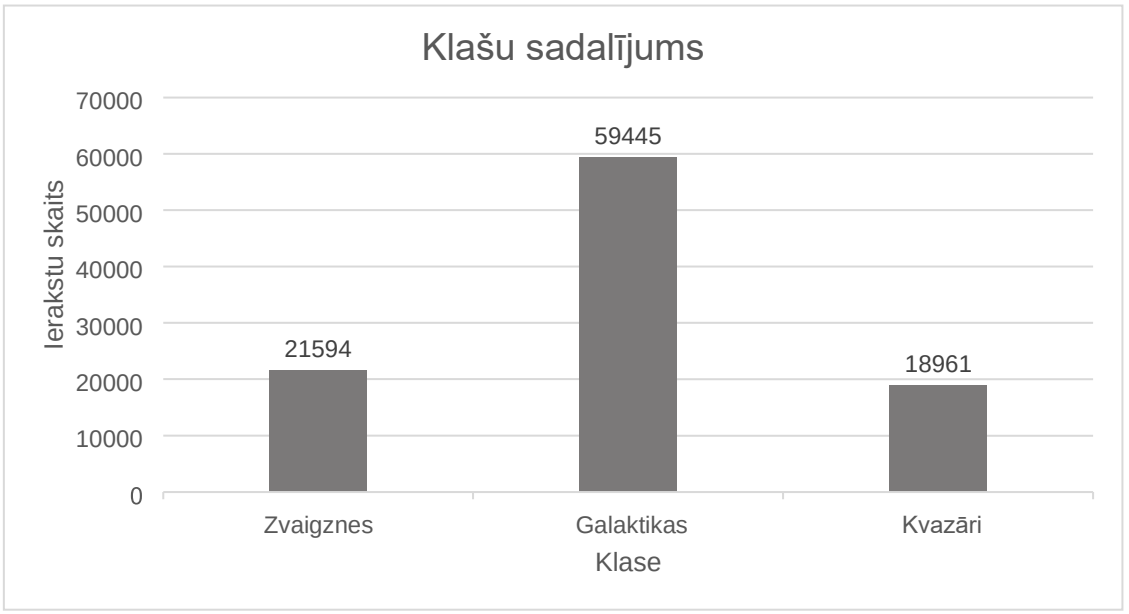
	obj_ID	alpha	delta	u	g	r	i	z	class	redshift
2	1237660961327743232	135.6891066036	32.4946318397087	23.87882	22.2753	20.39501	19.16573	18.79371	2	0.6347936
3	1237664879951151360	144.826100550256	31.2741848944939	24.77759	22.83188	22.58444	21.16812	21.61427	2	0.779136
4	1237660961330430208	142.1887895562506	35.5824441819976	25.26307	22.66389	20.60976	19.34857	18.94827	2	0.6441945
5	1237663478724297984	338.741037753146	-0.402827574587482	22.13682	23.77656	21.61162	20.50454	19.2501	2	0.9323456
6	1237680272041378048	345.282593210935	21.1838656010284	19.43718	17.58028	16.49747	15.97711	15.54461	2	0.1161227
7	1237680272039609088	340.995120509191	20.5894762801019	23.48827	23.33776	21.32195	20.25615	19.54544	3	1.424659
8	1237678858481565952	23.2349264301638	11.4181876197835	21.46973	21.17624	20.92829	20.60826	20.42573	3	0.5864546
9	1237678858473963776	5.43317603738404	12.0651859913473	22.24979	22.02172	20.34126	19.48794	18.84999	2	0.477009
10	1237661435386659840	200.290475389797	47.199402322911	24.40286	22.35669	20.61032	19.4649	18.95852	2	0.660012
11	1237670961088168192	39.1496905996484	28.1028416109607	21.74669	20.03493	19.17553	18.81823	18.65422	1	-7.895373e-06
12	1237680272034169856	328.092076173419	18.2203104791579	25.77163	22.52042	20.63884	19.78071	19.05765	2	0.4595958
13	1237662341088150272	243.986637469699	25.7382804319961	23.76761	23.79969	20.98318	19.80745	19.45579	2	0.5914091
14	1237680507721220352	345.801874402853	32.6728678500872	23.17274	20.14496	19.41948	19.22034	18.89359	1	7.182029e-05
15	1237678858459349248	331.502029984917	10.0358020468494	20.8294	18.75091	17.51118	17.01631	16.62772	2	0.1521936
16	1237663478726984960	344.984770271278	-0.352615781151814	23.20911	22.79291	22.08589	21.86282	21.8512	2	0.8181597
17	1237662341088543744	244.824523050208	25.1545639915034	24.8868	22.13311	20.44728	19.49171	18.9747	2	0.4849288
18	1237678598087508224	353.201522444633	3.08079593630972	24.5489	21.44267	20.95315	20.7936	20.48442	1	-0.000428576
19	1237678598091112704	1.494388639357	3.29174632998873	20.38562	20.40514	20.29996	20.05918	19.89044	3	2.031528
20	1237678598096748800	14.3831352206597	3.21432619593864	21.82154	20.5573	19.94918	19.76057	19.55514	1	-0.0004402762
21	1237651539783057664	167.131668785257	67.3399356293198	20.48292	18.67807	17.6168	17.11936	16.73351	2	0.1115879
22	1237651539783844096	171.975424574048	67.747450140585	22.13367	20.84772	18.96537	18.31696	17.98124	2	0.3747563
23	1237657589775073536	144.785292662052	46.8264956757313	24.54793	22.33601	20.92259	19.87177	19.16934	1	-0.0001203588
24	1237657589775204864	145.273037350992	46.9601338072329	25.44243	20.77028	19.6617	19.08481	18.83176	2	0.6623096
25	1237657589775401216	145.883005500431	47.300483575273	21.73992	21.53095	21.26763	21.36257	21.15861	3	2.07568
26	1237662341086970112	241.426267237893	27.2246949401119	18.88323	17.54229	17.01789	16.75376	16.72259	2	0.03208113
27	1237658423542022144	132.922468009168	4.52186469749673	21.2611	20.50495	18.36379	23.17828	17.96264	2	0.2509563
28	1237658423546544640	143.288017625325	5.55205221134132	25.98497	21.31456	19.61107	18.83178	18.27728	2	0.4612782
29	1237658423545364736	140.600038144857	5.26575770443853	25.46577	22.4065	21.43408	20.26256	19.98775	2	0.6110625
30	1237663478721872128	333.311510605612	-0.376122967149355	20.53324	18.84066	18.05369	17.60397	17.2903	2	0.09108476
31	1237663478723575808	337.093435465929	-0.311773269814488	20.15491	18.37295	17.31276	16.82294	16.44342	2	0.1482283
32	1237662341092606208	252.75854857538	19.4935268704245	24.36048	20.3777	18.53392	17.84004	17.44505	2	0.3846326
33	1237662341086380544	240.213908860967	27.9642908504061	23.08039	22.02426	20.80525	19.90149	19.37544	2	0.546072
34	1237668736824770816	255.574893510297	45.4787029190988	23.73066	22.82349	21.32414	19.81448	19.28439	2	0.7491816
35	1237678858480189696	20.0525557261385	11.4978807678312	21.89214	21.35124	21.18755	20.843	20.7658	3	1.528308
36	1237673709872218624	144.721737004651	5.84684728531905	22.88916	21.63309	20.06106	19.13263	18.9481	2	0.5091722
37	1237678439702987264	44.9233602672725	3.17554875120939	23.8628	22.7784	20.98458	19.70143	19.24533	2	0.5981273
38	1237654606389051904	136.263071151474	3.94400308209486	24.06677	23.00891	21.0967	20.12869	19.68312	2	0.5215806
39	1237657401346294016	140.512982663685	45.4036126655435	20.71552	18.85877	17.72017	17.24668	16.87722	2	0.1769632
40	1237660635987836928	136.418378205442	36.1526785230781	20.66654	22.21825	24.8026	21.43702	22.82647	1	0.0003084856
41	1237673705043460864	121.610009731402	41.9287206884987	22.42006	21.26692	19.40369	18.74999	18.26678	2	0.349364
42	1237662194525405440	181.645330520821	42.2739952211876	21.20149	19.77107	19.27176	19.04226	18.9441	1	4.827565e-05
43	1237671125376500480	146.925784211642	14.8800374884817	25.52618	22.45513	21.82048	20.83807	20.28395	2	0.8072085

(att.1)

Šajā tabulā (att.1) redzams datu kopas fragments, ar visām kolonnām, pēc datu modifikācijas veikšanas. Visas vērtības ir saglabātas kā reāli skaitļi - datu tipā long, izņemot klases atribūtu, kas saglabāts kā integer.

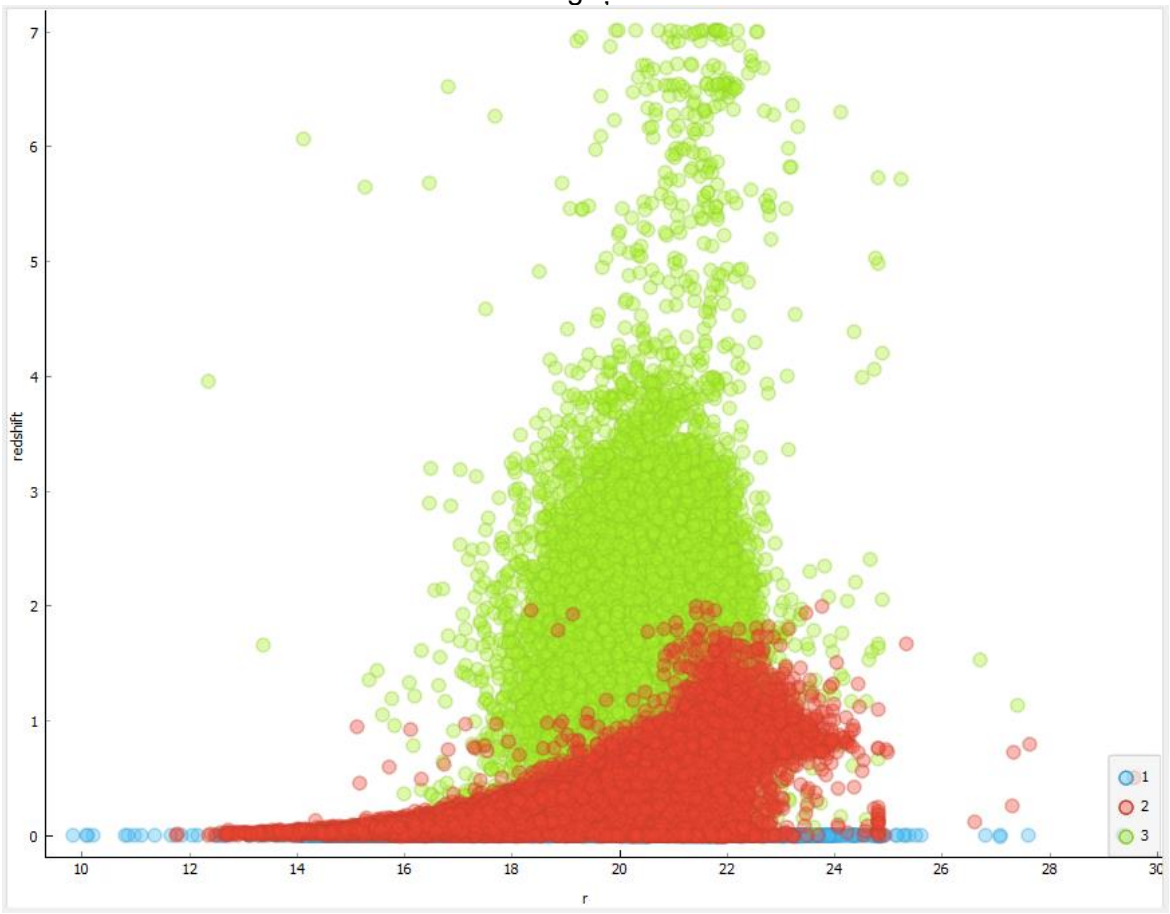
Aprēķinot minimālās vērtības un dispersiju, atklāju, ka viens ieraksts saturēja vērtību ‘-9999’ pie gaismas filtru vērtībām, tā kā šīs vērtības tiek izteiktas nanometros, šīs vērtības uzskatu par nederīgām un šo ierakstu izņemu ārā no datu kopas.

Datu kopas vizualizācijas



(att.2)

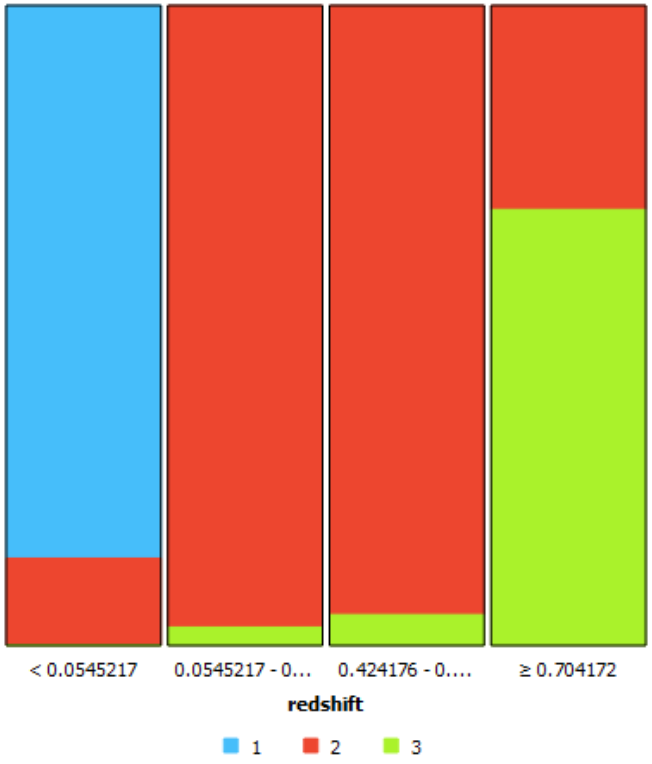
Šajā histogramā (att.2) var redzēt sadalījumu ierakstu skaitam katrai klasei – skaidri redzams, ka Galaktikas ir ~3 reizes vairāk kā kvazāru vai zvaigžņu.



(att.3)

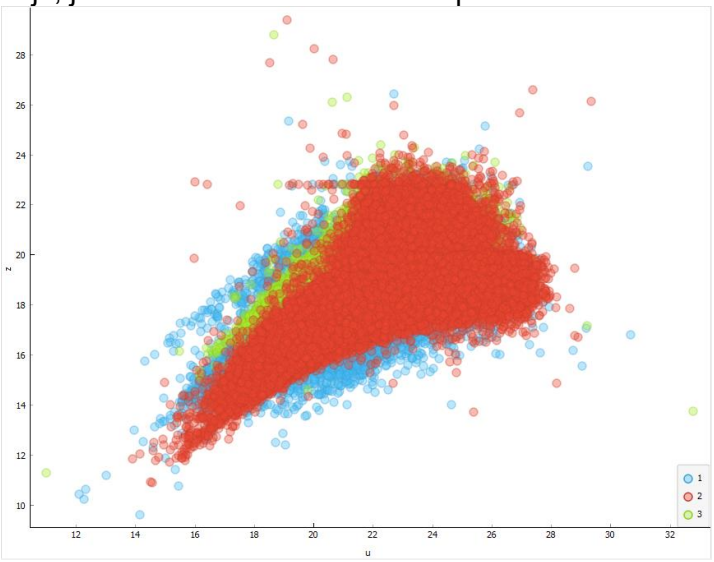
Diagrammā (att.3) var redzēt datu kopas izkliedes diagrammu, kas atspoguļo datu sadalījumu sarkanās gaismas nobīdes salīdzinājumam ar sarkanā filtra vērtību. Zaļās vērtības ir kvazāri,

sarkanās – galaktikas un zilās – zvaigznes. Varam novērot, ka lielākā sarkanā nobīde piemīt kvazāriem un gandrīz nekāda nobīde nav zvaigznēm. Šis ir izskaidrojams ar faktu, ka sarkanā nobīde palielinās, pieaugot gaismas ceļotajai distancei – tāpēc ka lielākā daļa novērotu zvaigžņu būs mūsu galaktikā, bet kvazāri būs ārpus tās, varam secināt, ka tiem būs lielāka sarkanā nobīde. To pašu varam redzēt apskatot galaktiku datus – ir galaktikas kas atrodas tuvu mūsu galaktikai, bet tālāk esošas galaktikas cietīs ar lielāku sarkano nobīdi. Diagramma (att.4) ilustrē šo pašu sakarību.



(att.4)

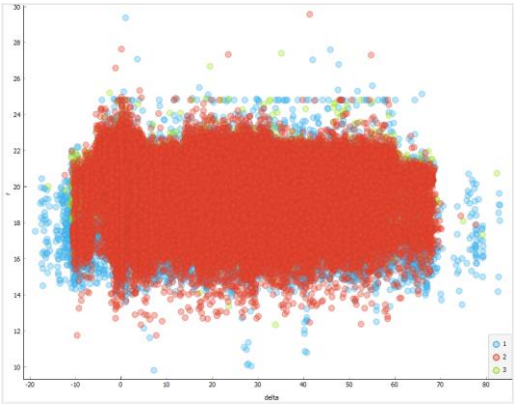
Redzam, ka zvaigznes ar sarkano nobīdi virs 0.054 neeksistē, savukārt lielākā daļa kvazāru, šī nobīde ir virs 0.7. Galaktikas atrodas pārsvarā pa vidu. Lai gan šis parametrs mums palīdz klasificēt mūsu datus, tas nav galējs, jo mēs nevaram klasi secināt pārliecināti balstoties uz to.



(att.5)

Izkliedes diagrammā (att.5) redzam 2 citu atribūtu savstarpējo sakarību – ultravioletās un infrasarkanās gaismas filtru vērtības – lai gan tās nav sagrupētas 3 vienmērīgās daļās, tās ir koncentrējušas noteiktos apgabalos, varam novērot, ka ir savstarpēji liela pārklāšanās.

Apskatot dažādas diagrammas, redzu, ka sarkanā nobīde ir vislabākais atribūts datu klasifikācijai. Ir arī parametri, kas mums nepalīdz atrast likumsakarības starp dažādām klasēm, šeit redzams piemērs, kura vērtības ir patvaļīgas starp trīs klasēm:



(att.6)

Attēlā 6 ir atspoguļota attiecība starp sarkanā gaismas filtra un deklinācijas leņķi – varam novērot ka dati katrā no klasēm ir atrodami visos deklinācijas leņķos un ar visiem sarkanā gaismas filtra vērtībām.

Name	Distribution	Mean	Median	Dispersion	Min.	Max.	Missing
obj_ID		1.23766e+18	1.23766e+18	6.81801e-06	1.23765e+18	1.23768e+18	0 (0%)
alpha		177.629	180.901	0.54328	0.00552783	360	0 (0%)
delta		24.1356	23.6465	0.813924	-18.7853	83.0005	0 (0%)
u		22.0806791	22.17914	0.1019469	10.99623	32.78139	0 (0%)
g		20.6315833	21.09993	0.0987502	10.49820	31.60224	0 (0%)
r		19.6457769	20.12531	0.0944098	9.82207	29.57186	0 (0%)
i		19.0849	19.4051	0.0921092	9.4699	32.1415	0 (0%)
z		18.769	19.0046	0.0940899	9.61233	29.3837	0 (0%)
redshift		0.576667	0.424176	1.26712	-0.00997067	7.01124	0 (0%)
class			2	0.955			0 (0%)

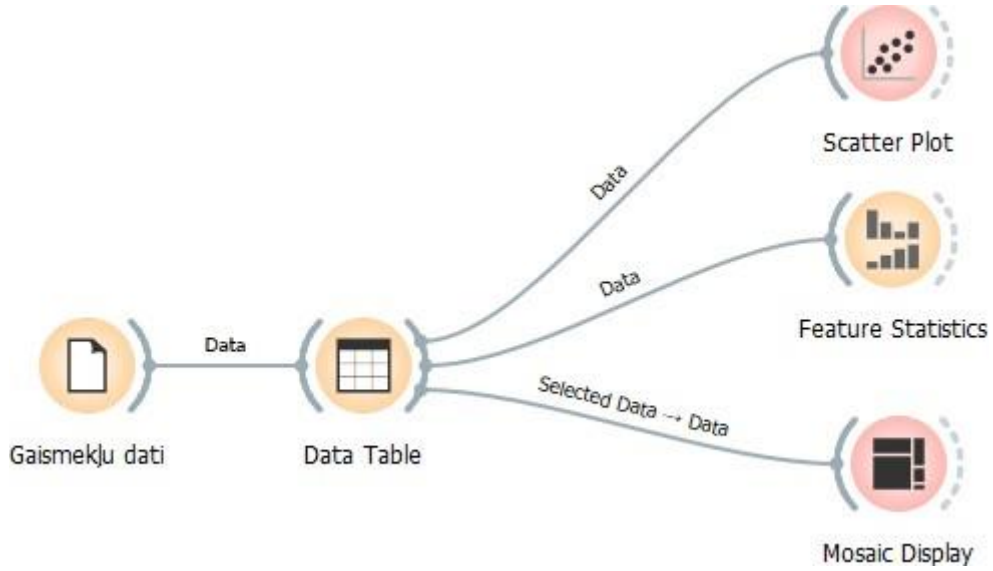
(att.7)

Izmanotojot logrīku ‘Feature Statistics’ iekš rīka Orange, varam redzēt informatīvas datu pazīmes. Redzam, ka nevienam no ierakstiem neviens no atribūtiem nav tukšs. Redzam, katram atribūtam vidējās, maksimālās un minimālās vērtības kā arī dispersiju katram atribūtam. Apskatot tabulu, var secināt ka atribūti ir dažādi – ir atribūti, kas ir ar ļoti mazu dispersiju un ir atribūti ar ļoti lielu dispersiju.

No visiem iepriekš pieminētajiem datu izpētes diagrammām secinu, ka nav konkrētu atribūtu, kas mums varētu ar pārliecību noteikt gaismas avota klasi. Mums ir atribūti, kuri nepalīdz klasificēt datu ierakstus(t.i. deklinācijas leņķis, sarkanās gaismas filtra vērtīgas). Mums ir vairāki atribūti kuri izveido nepilnīgu klasifikācijas iespēju (t.i. sarkanā nobīde, ultravioletās/infrasarkanās gaismas vērtības).

Es ceru, ka apvienojot mūsu nepilnīgos, bet informatīvos atribūtus un minimizējot to atribūtu svarus, kuri mums nepalīdz klasificēt datus, iespējams, mēs varētu secināt objekta klasi ar lielu pārliecību.

Pētot datus, lietoju Orange rīku un galā sanāca izveidot struktūru (att.8), kas izskatās minimāla, bet palīdzēja atrast daudz interesantu īpašību par datu kopu. Es apskatīju arī citus logrīkus, bet attēlā redzamie man izradījās par visnoderīgākajiem.



(att.8)

II daļa – Nepārraudzītā mašīnmācīšanās

Turpinot datu izpēti to izmantošu ‘k-Means’ un ‘Hierarchical Clustering’ algoritma logrīkus. Tāpēc ka manis izvēlētais datu sets ir lielāks, nekā to spēj izmantot šie logrīki, es izmantošu randomizētu datu setu ar 5 tūkstoš ierakstiem no kopējā datu seta, lai to izveidotu izmantošu logrīku ‘Data Sampler’.

k-Means logrīks

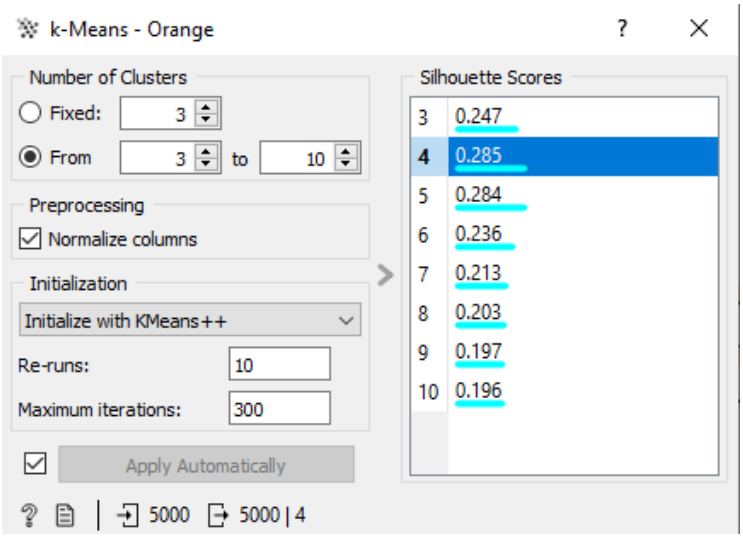
k-Means logrīks atļauj mums atspoguļot siluetu koeficientu dažādam skaitam klasteru. Tas satur trīs parametrus, ko varu modificēt, lai izmainītu siluetu koeficientu vērtības. Parametru skaidrojumi:

‘Number of Clusters’ – Definē skatu klasteriem, kam programma mēģina izveidot siluetu koeficientus.

‘Preprocessing’ – Ja opcija ir izvelēta, datu sets tiks normalizēts.(Visos mēģinājumos tiks izvēlēta)

‘Initializaiton’ – Nosaka kā centroidi tiks inicializēti. Tos var nejauši izvēlēt vai balstoties uz vidējām vērtībām. ‘Re-runs’ atribūts nosaka cik reizes algoritms atkārtosies un atribūts ‘Maximum iterations’ nosaka cik iterāciju būs katrā algoritma izpildes reizē.

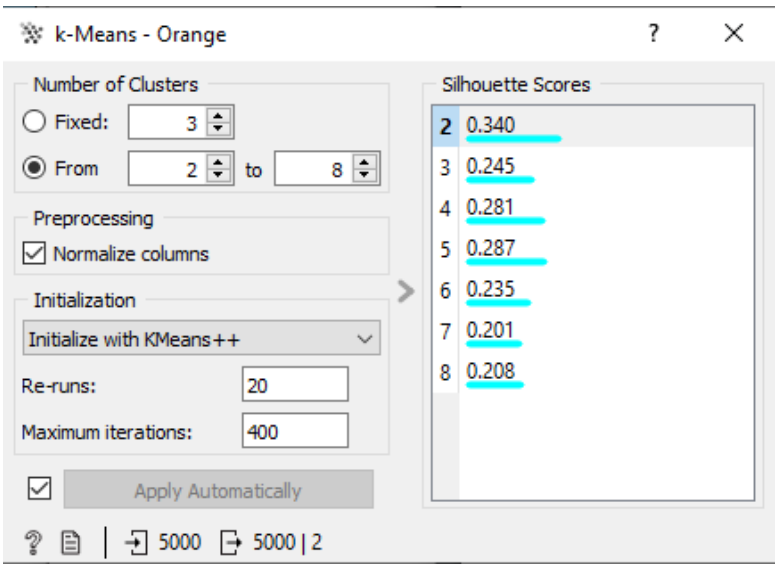
Pirmajā mēģinājumā parametrā ‘Number of Clusters’ nosaku, lai tiktu izvadītas 3 līdz 10 klasteru siluetu koeficienti. Inicializēju ar KMeans++, 10 atkārtojumi algoritmam un maksimums 300 iterācijas katrā izpildē.



(att.9)

Kā redzams (att.9), algritmam nav iespējams identificēt 3 skaidras klases no dotajiem ievaddatiem. K-Means algoritms saka, ka dati vieglāk klasificējami 4 vai 5 klasēs nekā 3, kā tas ir dots mūsu datu kopā. Kopumā siluetu salīdzinājumi ir tuvi viens otram, izņemot 9 vai 10 klasterus, kuru koeficients jau ir zem 0.2.

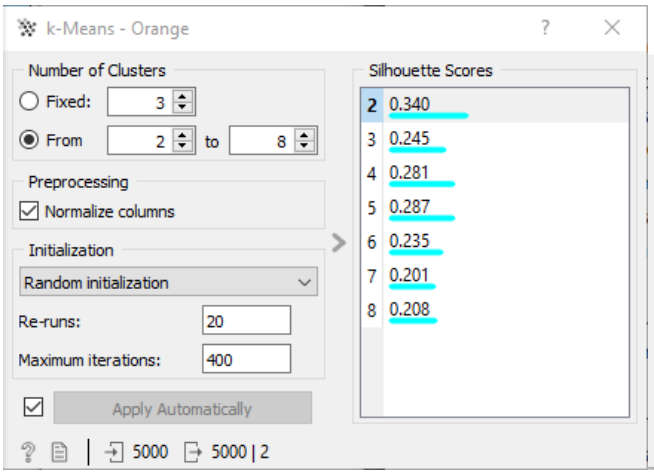
Otrajā mēģinājumā samazināšu klasteru skaitu uz diapazonu 2-8, palielināšu maksimālo iterāciju skaitu uz 400 un algoritma atkārtojumu skaitu uz 20:



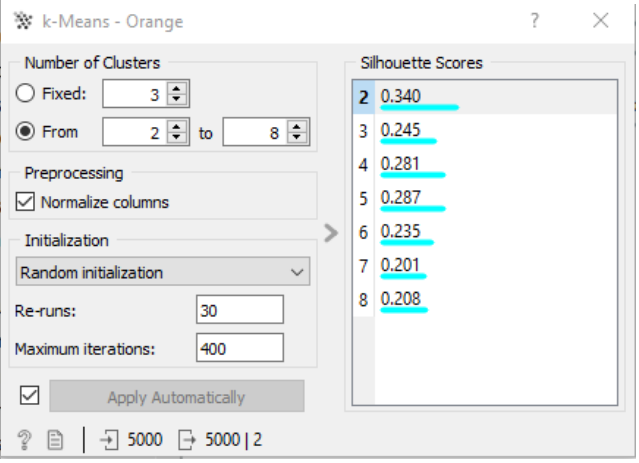
(att.10)

Kā redzam šeit izvadīto siluetu koeficienti atšķirās – visticamākais sadalījums ir 2 klasteri, 2 nākamie ir 4 un 5. Šis rezultāts ir interesants, jo patiesais 3 klasteru sadalījums ir mazāk ticams nekā 2, 4 vai 5.

Izmainot Inicializācijas metodi un randomizētu inicializāciju, attēlā 11 redzam ka rezultāti nemainās. Palielinot algoritma izpildes atkārtojumu skaitu, varam novērot, ka rezultāti nemainās, tas liek domāt, ka algoritms ir nostabilizējies un izmaiņas siluetu koeficientos vairs nenotiek.



(att.11)



(att.12)

No šiem rezultātiem secinu, ka izmantojot k-Means algoritmu, mēs nevaram atrast 3 klasteru sadalījumu, kas atbilstu vairāk, kā atbilstu 2, 4 vai 5 klasteru sadalījumi.

Hierarhiskā klasterizācija

Logrīks ‘Hierarchical Clustering’ jāizmanto kopā ar logrīku ‘Distances’, kas izveido attālumus starp kolonnām un ierakstiem datu kopā. Ar šo logrīku iespējams grupēt satu kopas ierakstus izmantojot hierarhiskās klasterizācijas algoritmu. Tā pat kā k-Means logrīks, tas satur parametrus, ko varu modificēt, lai izmainītu siluetu koeficientu vērtības. Parametru skaidrojumi:

‘Linkage’ – Definē datu saistīšanas metodi.

‘Single linkage’ – Aprēķina distance starp tuvākajiem elementiem divos klasteros.

‘Avergae linkage’ – Aprēķina videjo distance starp elementiem divos klasteros.

‘Weighted linkage’ – Izmanto WPGMA metodi lai apreķinātu distance.

‘Complete linkage’ – Aprēķina distanci starp tālākajiem elementiem divos klasteros.

‘Ward linkage’ - Aprēķina distanci starp elementiem divos klasteros samazinot kopējo distanču varianci.

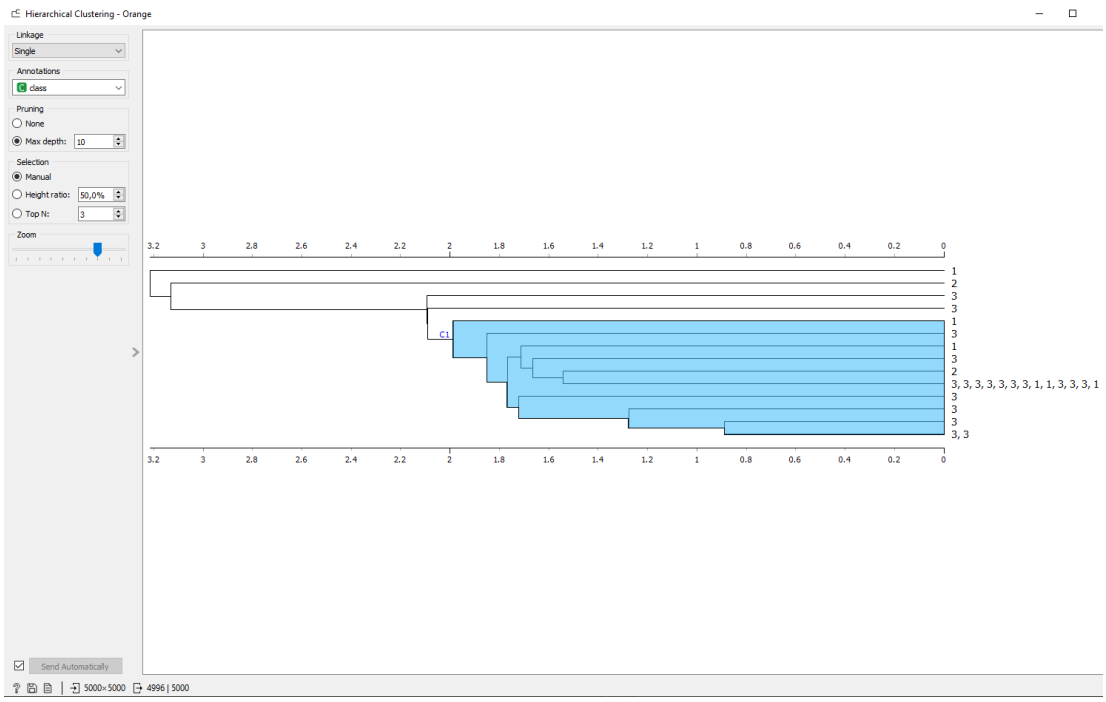
‘Annotations’ – Definē kuras kolonnas datas atpoguļot skatā.

‘Prunning’ – Definē dendrogrammas maksimālo dziļumu.

‘Selection’ – Definē, kurā vietā dendogramā tiks atdalītas klases.

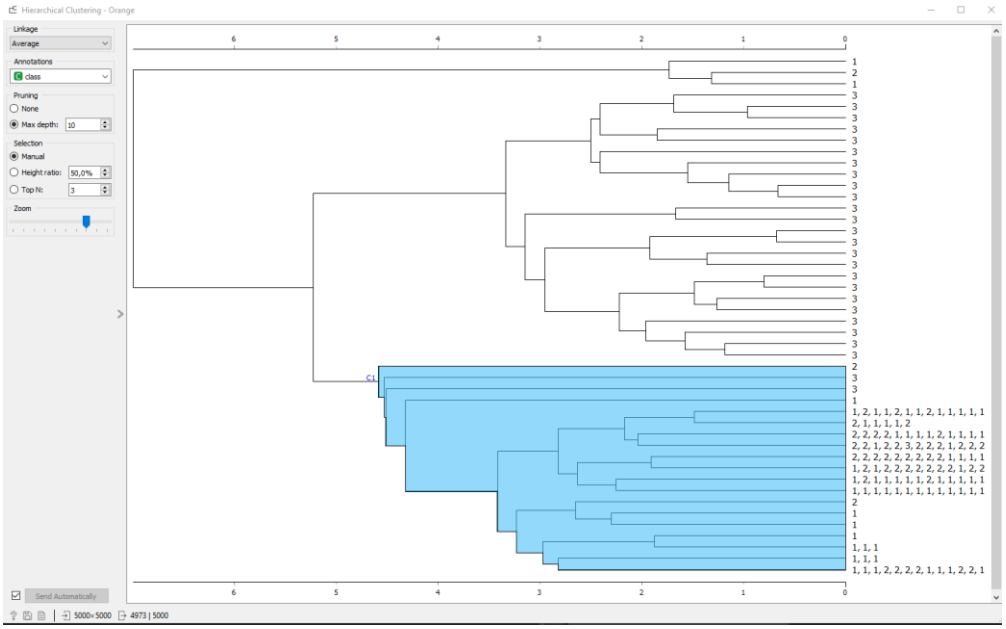
**Skaidrojumi iegūti no Orange rīka, Help sadaļas.*

Lai salīdzinātu dažādu šo aprakstīto hiperparametru darbības, definēšu kostantu dendrogrammas maksimālo dziļumu (Prunning) kā 10. Un anotācija visām dendrogramām saturēs klases kolonnas vērtību.



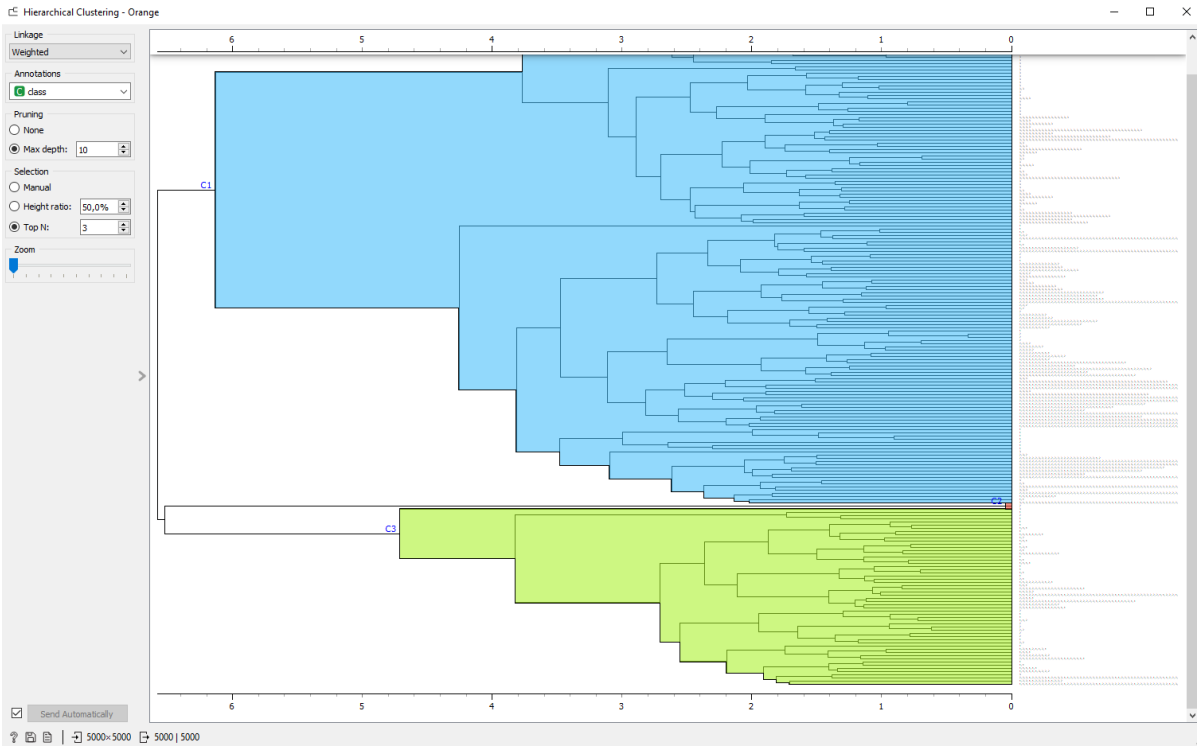
(att.13)

Dendrogramā att.13 varam novērot pirmo mēģinājumu sagrupēt datu kopu, to daru izmantojot ‘Single linkage’ sasaistīšanas metodi – kā var redzēt, tai ir izdevies sasaistīt 2 grupas katrā pa diviem elementiem un trešajā grupā ir ievietoti 4996 elementi. Salīdzinot šo rezultātu ar attēlā 14 redzamo rezultātu, kura ir izmantota ‘Average linkage’ sasaistīšanas metode, varam secināt, ka vidējo distanču salīdzināšana ir efektīvāka, jo tā ir sagrupējusi vairāk elementu iekš klasēm – 3, 24 un 4973, taču jāsaka ka arī šads rezultāts nav gluži pieņemams klasifikācijas programmai.



(att.14)

Apskatot ‘Weighted linkage’ sasaistīšanas metodi varam redzēt ka vel vairāk ierakstu ir sasaistīti – 1 grupa kurā ir 1 galaktika, otra grupa kurā ir 523 zvaignes un galaktikas un atlikušie 4476 ieraksti ielikti trešajā grupā.

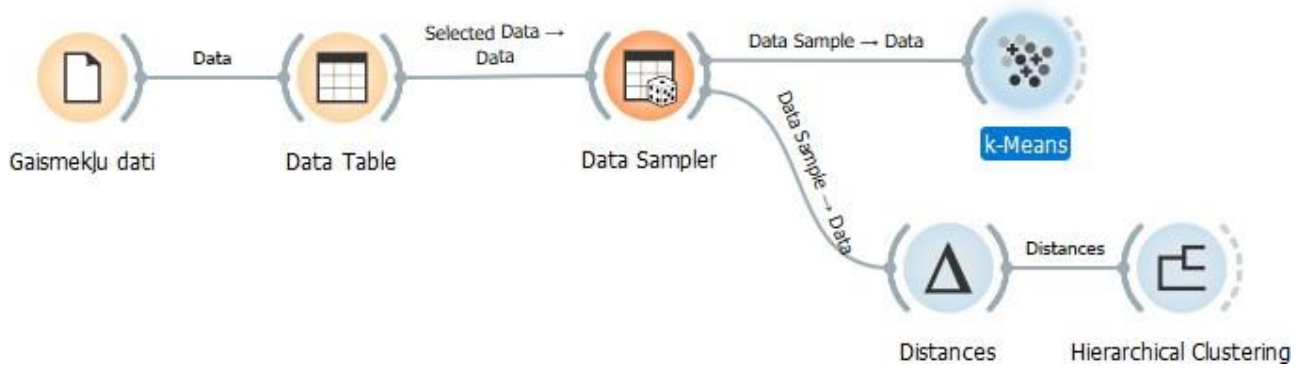


(att.15)

Apskatot ‘Complete linkage’ un ‘Ward linkage’ sasaistīšanas metodes var redzēt ka izveidojas daudz vienmērīgāka datu sagrupēšana. Detalizēti apskatot vērtības kas ir iekļautas katrā no klasēm var redzēt, ka ‘Complete linkage’ metodē, algoritmam ir izdevies sagrupēt vienā grupā tikai kvazārus, otrā grupā tikai zvaigznes un galaktikas, bet trešaja var redzēt visu trīs klašu piederīgos elementus. Savukārt ‘Ward linkage’ metodē izdevies sagrupēt vislīdzīkāgā izmēra (skaita ziņā) klases, bet apskatot ierakstus tajās, varam redzēt katrai no mūsu 3 klasēm piederošos elementus.

No hierarhiskās klasterizācijas metožu salīdzinājuma secinu, ka ‘Complete linkage’ ir darbojies visprecīzāk, bet neviena no metodēm nav bijusi pilnīga datu klasterizācijai.

Kopumā, lai izveidotu nepārraudzītas mašīnmācīšanās klasterizāciju izveidoju šādu (att.16) struktūru rīkā Orange.



(att.16)

III daļa – Pārraudzītā mašīnmācīšanās

Lai izveidotu pārraudzītas mašīnmācīšanās modeli, sākumā ir jāizveido testa un treniņu datu kopas, to var izdarīt izmantojot logrīku 'Test and Score'. Šis logrīks izveido 2 datu kopas, vienu, kas saturēs datus ar ko trenēt modeli un otru ar ko pārdbaudīt tā precizitāti. Lai to paveiktu, logrīks piedāvā vairākas iespējas kā to izdarīt un tam ir vairāki hiperparametri, kas ļauj dažādos veidos izveidot šīs divas datu kopas. Parametru skaidrojumi:

'Sampling' – Definē kā sadalīt datu kopu starp testēšanas datiem un trenēšanas datiem.

> 'Cross validation' – Sadala datu kopu noteiktā skaitā daļu un šīs daļas tiek izmantotas gan testēšanai gan trenēšanai repteatīvi.
>
> 'Random sampling' – Sadala datu kopu divās daļās noteiktā procentu sadalījumā nejauši izvēloties kurus ierakstus liekot kurā daļā.
>
> 'Leave-one-out' – Izveido modeli ar visiem datiem izņemot vienu un klasificē šo vienu izlaisto, atkārtojot šo darbību visiem elementiem.
>
> 'Test on train data' – Izmanto visu datu kopu priekš trenēšanas un testēšanas.
>
> 'Test on test data' – Izmanto atsevišķu testa datu failu.

**Skaidrojumi iegūti no Orange rīka, Help sadaļas.*

Priekš šī praktiskā sarba izvēlos lietot 'Cross validation' ar 5 daļām(folds).

Lai varētu izmantot logrīku 'Test and Score', tam vajag arī pievienot modeļu logrīkus kā ievaddatus, lait tas saprastu, kur izmantot šos testa un trenēšanas datus. Kā pārraudzītās mašīnmācīšanās modeļus izvēlos 'Random Forest' un 'kNN'.

Random Forest

Random forest modelēšanas logrīks ļauj veidot 'Random forest' mašīnmācīšanās algoritmu, datu kopas analizēšanai. Algoritmu izveidoja Tin Kam Ho 1995. gadā. To izmanto priekš regresijas, klasifikācijas un citiem uzdevumiem. Šis algoritms strādā pēc sekojošā principa – tas izveido vairākus izvēles kokus (decision tree), tie satur nejauši izvēlētas apakškopas no datu kopas, kas tiek apstrādāta.[1] Katrā koka lapā tiek nejauši izvēlēti kādi atribūti, kas tiek novērtēti un kādi, kas tiek ignorēti, lai veiktu koka sadalīšanu.[2] Algoritma rezultātā tiek izveidots kāds skaits izvēles koku un tiem netiek pievienoti svari - algoritma rezultāts ir iegūts apvienojot visu izvēles koku rezultātus un nosakot kura vērtība ir izvēlēta visbiežāk. Izvēlos šo algoritmu jo tas ir ļoti populārs un balstoties uz izmēģinājuma datu analīzēm sniedz ļoti precīzus rezultātus. Šis algoritms satur vairākus hiperparametrus, kurus mainot, varam ietekmēt tā darbību. Parametru skaidrojumi:

'Number of trees' – Definē cik daudz izvēles kokus(decision trees) iekļaut mežā(forest).

'Number of trees considered at each split' – Definē cik datu kopas atribūti būs nejauši iekļauti katrā koka virsotnē.

'Replicable training' – Definē, vai saglabāt koku atkārtotam algoritma izpildījumam.

'Balance class distribution' – Definē, vai svaru klases ir inversi proporcionālas to frekvencēm.

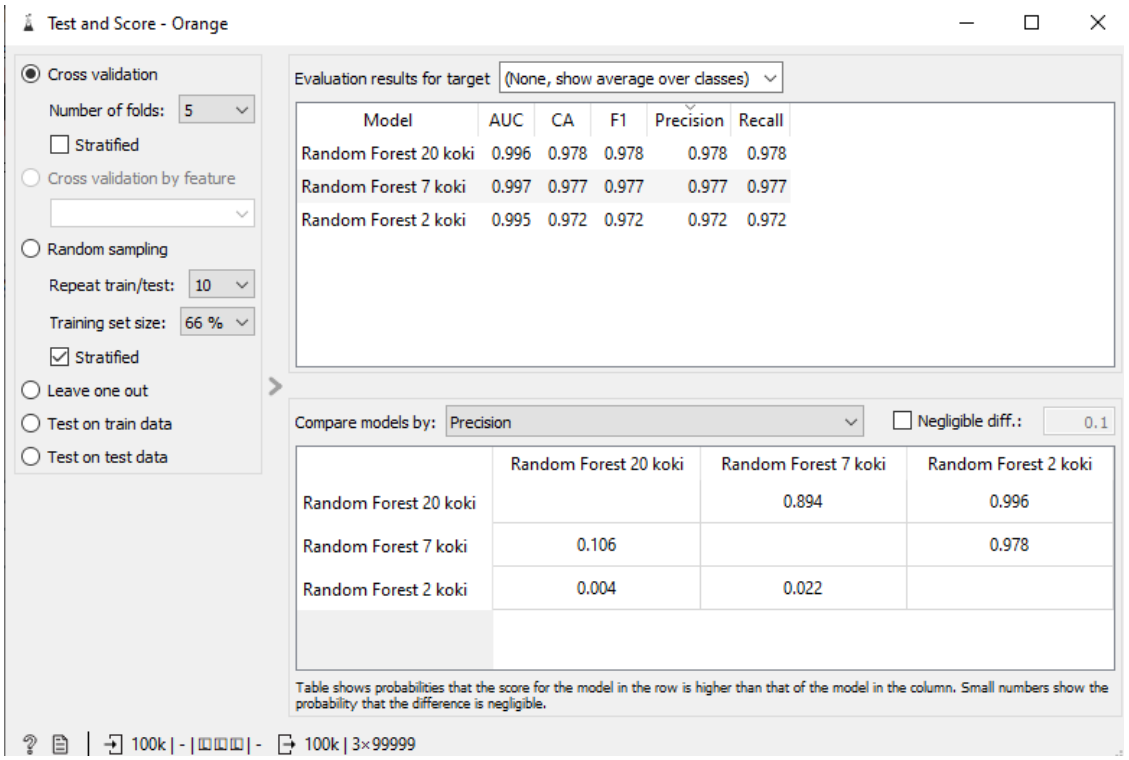
‘Limit of depth of individual tress’ – Definē maksimālo koka dziļumu.

‘Do not split subsets smaller than’ – Definē minimālo elementu skaitu koka virsotnē.

**Skaidrojumi iegūti no Orange rīka, Help sadaļas.*

Lai salīdzinātu rezultātus apskatīšu klasifikācijas precizitāti(Precision), kas pasaka cik precīzi modelis paredz elementa klasi balstoties uz ievaddatiem. Klasifikācijas precizitāti mēra no 0 līdz 1 – jo lielāka vērtība, jo modelis pareizāk paredz elementa klasi.

Pirmajā mēģinājumā salīdzināšu koku daudzuma skaita ietekmi uz klasifikācijas precizitāti. Definēju minimālo datu elementu skaitu virsotnē – 5 un izvēlos 3 dažādas vērtības priekš koku daudzuma – 2, 7 un 20. Es izveidoju šos 3 ‘Random forest’ modeļus un izvadu datus ‘Test and Score’ logrīkā. Attēlā 17 rezultāti ir sakārtoti pēc klasifikācijas precizitātes vērtības no lielākās uz mazāko.



(att.17)

Kā varam redzēt attēlā 17, lai gan rezultātu atšķirība ir maza un tikai izmantoti 3 dažādi koku daudzumi ir izveidojies trends – vairāk koku padara modeli precīzāku.

Lai redzētu citu hiperparametru ietekmi uz modeli un galā iegūtu visprecīzāko algoritmu, pārējiem mēģinājumiem atstāju koku daudzumu 20. Tālāk novēroju vai svaru klasēm esot inversi proporcionālām to frekvencēm uzlabo rezultātu. Izveidoju 2 modeļus – vienam atķeksējot lai šīs vērtības ir inversas, otram nē. Uzreiz programma ziņo, ka padarot vērtības inversas, modeļa aprēķināšana prasa vairāk laiku. Tas ir tāpēc, ka ir nepieciešamas papildu kalkulācijas algoritma izpildē. Taču apskatot rezultātus, atšķirību tajos nevar novērot (att. 18)

Model	AUC	CA	F1	Precision	Recall
Random Forest Ir Inversas	0.996	0.978	0.978	0.978	0.978
Random Forest Nav Inversas	0.996	0.978	0.978	0.978	0.978

(att.18)

Man gan apskatot šos rezultātus radās aizdomas, ka modelim jau funkcionējot ļoti efektīvi pateicoties koku skaitam, rezultātos nevar novērot atšķirību, tāpēc samazināju koku daudzumu un palielinot minimālo virsotnes elementu skaitu katram no kokiem. Veicot vairākas atkārtotas izpildes novēroju minimālu atšķirību algoritma rezultātā – dažreiz tas sliecās par 0.001 vienību labvēlīgāk inversām vērtībām, citreiz labvēlīgāk ne-inversām vērtībām. Tāpēc ka ietekme uz rezultātu ir tik maza un ietekme uz programmas skaitļošanas resursiem ir lielāka, izvēlos neatķeksēt šo hiperparametru.

Nākamais hiperparametrs kura ietekmi vēlos novērot uz modeļa izpildi ir minimālais elementu skaits koka virsotnēs – lai to izdarītu atkal veidoju 3 kokus ar dažādām šī hiperparametra vērtībām – 5, 50 un 500. Varam spriest, ka samazinoties šo elementu skaitam, būs nepieciešams izveidot dziļākus kokus, kas raisīs rezultātiem būt precīzākiem. To arī varam novērot attēlā 19 redzams, ka vērtības 5 un 50 ir precīzākas nekā 500.

Model	AUC	CA	F1	Precision	Recall
Random Forest min 5 elementi	0.996	0.978	0.978	0.978	0.978
Random Forest min 50 elementi	0.998	0.977	0.977	0.977	0.977
Random Forest min 500 elementi	0.998	0.973	0.973	0.973	0.973

(19.att.)

Adaboost

Modelēšanas logrīks AdaBoost ļauj veidot modeli ar AdaBoost mašīnmācīšanās algoritmu. AdaBoost algoritmu izstrādāja Yoav Freaund un Robert E. Schapire 1995. gadā (Yoav Freund Robert E. Schapire, 1999) Šo algoritmu dēvē par meta-algoritmu, jo tas apvieno vairāku algoritmu darbību. Algoritms ieņem treniņa datus un katram ierakstam piesaista svaru.[3] Sākumā visi svari ir vienādi, bet katrā raundā, nepareizi klasificēto piemēru svaru vērtības tiek palielinātas un pārējās svaru vērtības pārrēķinātas un normalizētas, lai algoritms vairāk fokusētos uz nepareizi identificētajām vērtībām. Katrā raundā tiek izvēlēts viens algoritms, kas vislabāk paredzēja vērtību iznākumus un tam tiek piesaistīts lielāks svars nekā citiem. Galu galā algoritms izveido vairākus modeļus, katru ar savu svaru, kad ievaddati tiek doti adaBoost modelim, tie tiek izvadīti caur katru no modeļiem un balstoties uz katra modeļa svaru, tiek noteikts modeļa 'meta' rezultāts[4].

Man šis algoritms ieintriģēja jo tas nebalstās uz vienu specifisku algoritmu, bet gan uz vairākiem savstarpēji saistītiem modeļiem – dzirdot ka mašīnmācībā var izmantot daudz dažādu modeļu, instinktīva liekas doma – kas, ja mēs apvienotu vairākus modeļus, vai varētu labāk paredzet vērtības? Otrs iemesls, kapēc izvēlos šo algoritmu ir, jo kad izmēģināju dažādus modeļus Orange rīkā, šis algoritms bija ļoti precīzs, tāpēc vēlos salīdzināt to ar 'Random forest' algoritmu un redzēt, kurš ir precīzāks.

Šis algoritms satur vairākus hiperparametrus, kurus mainot, varam ietekmēt tā darbību. Parametru skaidrojumi:

'Number of estimators' – maksimālais daudzums novērtētāju, kad algoritma rīcība tiek pārtraukta.

‘Learning rate’ – ātrums, ar kādu katrā iterācijā mainās svari.

‘Classification algorithm’ – Algoritms, pēc kura tiek veikta klasifikācija. ‘SAMME’ – Izmaina novērtētāju svarus pēc klasifikācijas rezultātiem.
‘SAMME.R’ – Izmaina novērtētāju svarus pēc varbūtības paredzējumiem.

‘Regression loss function’ – Nosaka regresijas funkciju **tiek izmantots pie regresijas uzdevumiem.*

Lai novērtētu kā hiperparametri ietekmē modeļa darbību, sākšu izmainot maksimālo novērtētāju skaitu – 25, 50, 75, 100. 50 tiek definēta kā *default* vērtība un apskatot rezultātus(att.20) varam redzēt kapēc – funkcijas efektivitāte nemainās, mainot šo hiperparametru.

Model	AUC	CA	F1	Precision	Recall
AdaBoost 25	0.986	0.965	0.965	0.965	0.965
AdaBoost 50	0.986	0.965	0.965	0.965	0.965
AdaBoost 75	0.986	0.965	0.965	0.965	0.965
AdaBoost 100	0.986	0.965	0.965	0.965	0.965

(att.20)

Nākamais hiperparametrs, ko apskatu ir ‘Learning rate’. Atkal izveidoju 4 modeļus, vienīgi izmainot šo parametru. Arī šis parametrs, atšķirības rezultātos neizmaina, kā redzams attēlā 21.

Model	AUC	CA	F1	Precision	Recall
AdaBoost learning rate .25	0.881	0.965	0.965	0.965	0.965
AdaBoost learning rate .5	0.881	0.965	0.965	0.965	0.965
AdaBoost learning rate .75	0.881	0.965	0.965	0.965	0.965
AdaBoost learning rate 1	0.881	0.965	0.965	0.965	0.965

(att.21)

Kā pēdējo atribūtu kā ietekmi uz modeli es apskatu ir ‘Classification algorithm’ – izveidoju 2 modeļus, vienam tiek izvmantots ‘SAMME’ otram ‘SAMME.R’ modelis. Tieši tā pat kā iepriekšējiem diviem hiperparametriem, nekāda atšķirība modeļa darbībā netiek konstatēta.

Model	AUC	CA	F1	Precision	Recall
SAMME.R	0.881	0.965	0.965	0.965	0.965
AdaBoost SAMME	0.881	0.965	0.965	0.965	0.965

(att.22)

Salīdzinājums starp Random Forest un AdaBoost.

Salīdzinot abus izvēlētos pārraudzītās mašīnmācīšanās algoritmus, varam redzēt, ka abi modeļi ļoti precīzi – ar 97+% precizitāti nosaka gaismas avotu balstoties uz testa datiem. Random Forest modelis to dara nedaudz precīzāk – par 1.3%.

Model	AUC	CA	F1	Precision	Recall
Random Forest	0.975	0.978	0.978	0.978	0.978
AdaBoost	0.881	0.965	0.965	0.965	0.965

(att.23)

Lai salīdzinātu abu algoritmu darbību vairāk, salīdzināšu to veiktspēju dažādu trenēšanas un testēšanas datu sadalījumos. Attēli 24 un 25 satur tabulas ar dažādu trenēšanas un testēšanas datu sadalījumiem un to respektīvajām precizitātēm. Iegūstot šos rezultātus izmantoju 10 reižu repetīciju trenēšanas datu sadalījumam.

AdaBoost

Trenēšanas dati, %	Testēšanas dati, %	Pareizi kateg., %
95	5	96.4
90	10	96.5
80	20	96.5
75	25	96.5
70	30	96.5
64	36	96.5
60	40	96.5
50	50	96.4
40	60	96.4
33	67	96.3
30	70	96.3
25	75	96.3
20	80	96.3
10	90	96.0
5	95	95.8

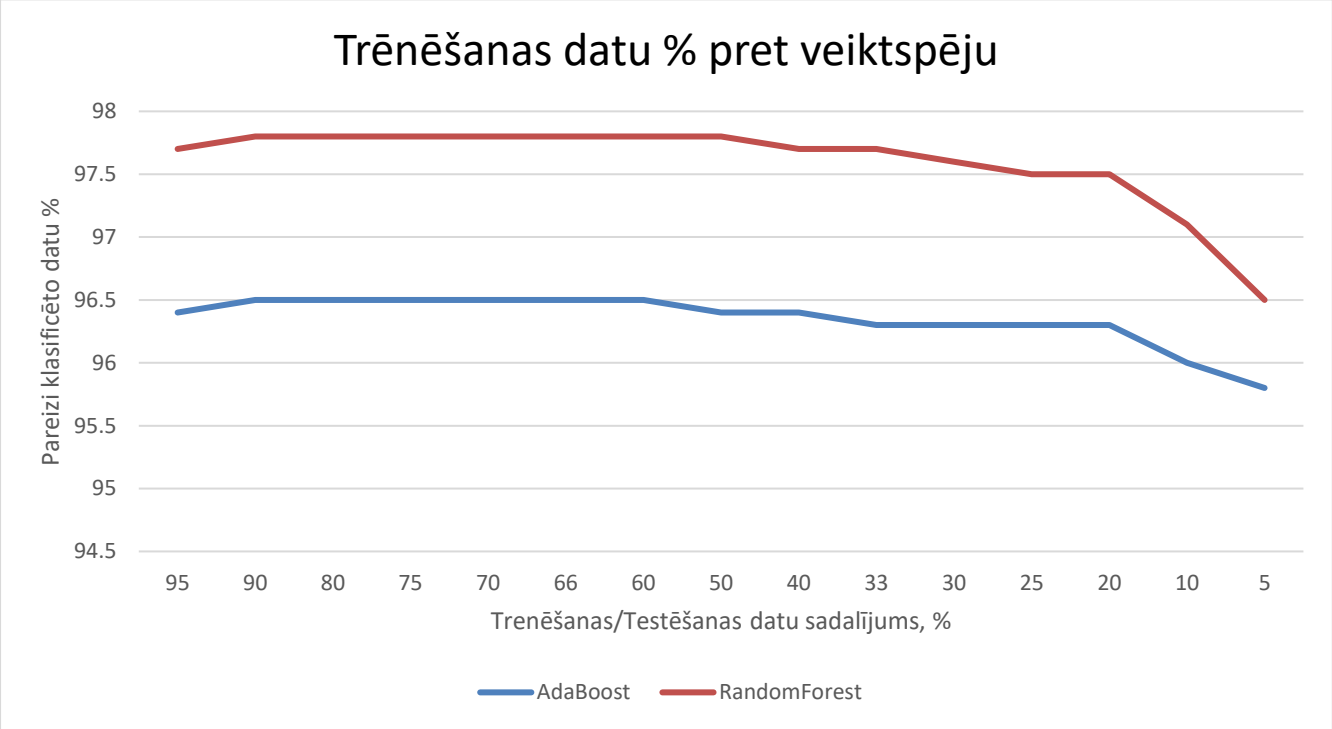
(att.24)

Random Forest

Trenēšanas dati, %	Testēšanas dati, %	Pareizi kateg., %
95	5	97.7
90	10	97.8
80	20	97.8
75	25	97.8
70	30	97.8
66	34	97.8
60	40	97.8
50	50	97.8
40	60	97.7
33	67	97.7
30	70	97.6
25	75	97.5
20	80	97.5
10	90	97.1
5	95	96.5

(att.25)

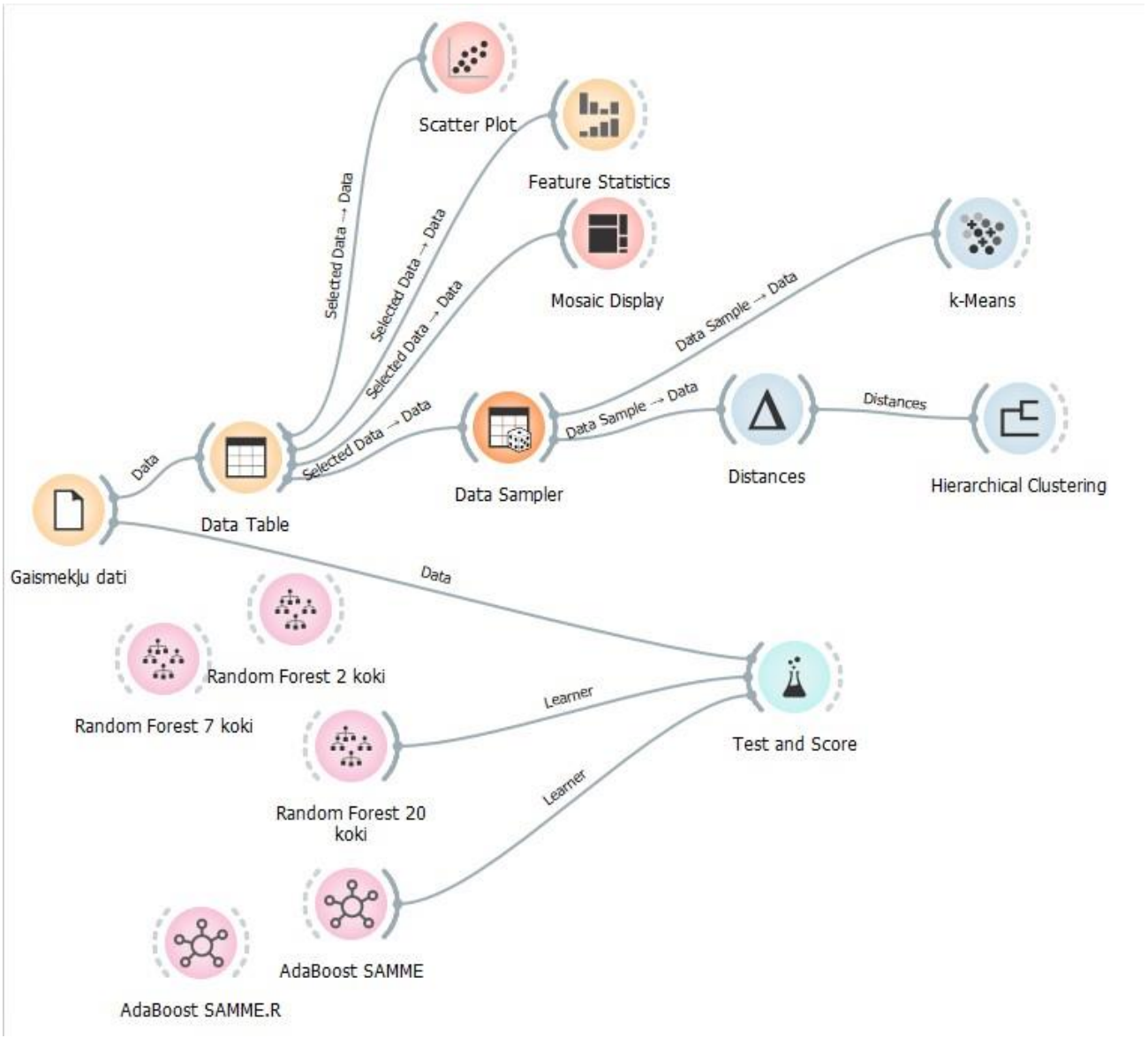
Šīs tabulas apvienojot var tikt izveidots grafiks(att.26), kas labāk ilustrē attiecību starp procentuālo trenēšanas datu izmantošanu un algoritmu veiktspēju. Kā redzam grafikā, palielinot trenēšanas datu procentuālo sadalījumu pret testēšanas datiem, algoritms uzlabojas, bet virzoties virs 60-40 procentu sadalījumam spēcīgas izmaiņas vairs nenotiek. Šos rezultātus var skaidrot ar spriedumu, ka trenēšanas datiem esot būtiski mazākiem par testēšanas datiem, testēšanas dati iekļauj parametru vērtības, kuras algoritmam nav nācies sastapt trenējoties. Taču, jāatdzīst, ka esmu pārsteigts, ka pat pie 5% trenēšanas datu, lai gan tas tika atkārtots 10 reizes, algoritmi ir virs 95% precizitātes.



(att.26)

Rezultātos varam novērot, ka, pareizi klasificēto datu % ir lielāks Random forest modelim nekā AdaBoost.

Kopējā Orange rīka darbplūsma



(att.27)

Secinājumi

Šī darba izpildes laikā es daudz iemācijos par datu ievākšanu, to apstrādi, nepārraudzītas un pārraudzītas mašīnmācīšanās modeļu izveidi un Orange rīka izmantošanu. Datu kopa ko izvēlējos likās ļoti piemērota šim uzdevumam, jo tā satūrēja pilnīgus datus un nebija jāveic daudz datu pārveide, lai tos kvalitatīvi izmantotu darba risināšanā. Daži no parametriem, kas tika izmantoti, kā es novēroju datu izpētes solī, nebija tik ietekmīgi kā citi un man būtu interesanti paskatīties kā to izkļaušana no datu seta atspoguļotots modeļu kvalitātē – vai tie tomēr kaut kādā mērā palīdz klasificēt informāciju, vai tomēr, pat ar maziem svariem pasliktina modeļu veiktspēju. AdaBoost un Random Forest algoritmi, kurus apskatiju detalizētāk, manuprāt ir ļoti spejīgi algoritmi, kurus izmantojot var iegūt ļoti kvalitatīvus pareģojumus – par to liecina ap 97% pareizi klasificēto ierakstu abiem algoritmiem.

Vispārsteidzošākais man likās Orange rīka izmantošanas ērtums – tas ļauj paveikt ļoti daudz darbību priekš datu kopas apstrādes, vizualizācijas un dažādu algoritmu izpildes. Man ļoti patika mācīties par un izmantot Orange rīku – tas izraisīja aizrautību par datu kopas izpēti un dažādo mašīnmācīšanās algortimu izmantošanu. Man šis rīks liekas ļoti intuitīvs un parocīgs, iepriekš man bija iespaids, ka lai izmantotu dažādus mašīnmācīšanās algoritmus ir nepieciešamas ļoti advancētas programēšanas zināšanas, bet šis rīks ļauj veidot daudz dažādu modeļu ar skaistu vizuālu interfeisu.

Izmantotā literatūra

[1] - Ho, Tin Kam (1995). Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282. Archived from the original (PDF) on 17 April 2016. Retrieved 5 June 2016. Pieejams: https://web.archive.org/web/20160417030218/http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf

[2] - Breiman L (2001). "Random Forests".Statistics Department, University of California, Berkeley, CA 94720, January 2001. Pieejams: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

[3] - Yoav Freund Robert E. Schapire, 1999. A Short Introduction to Boosting. AT&T Labs Research Shannon Laboratory 180 Park Avenue Florham Park, NJ 07932 US. Pieejams: https://cseweb.ucsd.edu/~yfreund/papers/IntroToBoosting.pdf

[4] - StatQuest with Josh Starmer, “AdaBoost, Clearly Explained”. Pieejams: https://www.youtube.com/watch?v=LsK-xG1cLYA