

**RĪGAS TEHNISKĀ UNIVERSITĀTE**

Datorzinātnes un informācijas tehnoloģijas fakultāte

2. praktiskais darbs

mācību priekšmetā

*“Mākslīgā intelekta pamati”*

**Mašīnmācīšanās algoritmu lietojums**

Izstrādāja: Jānis Lielkalns

St. apl. Nr. 201RDB203

Pārbaudīja:

Dr.sc.ing. Alla Anohina-Naumeca

**Saturs**

[I Daļa– Datu pirmapstrāde/izpēte 4](#_Toc104499445)

[Datu kopas apraksts 4](#_Toc104499446)

[Datu kopas satura apraksts 5](#_Toc104499447)

[I Daļa – Secinājumi 10](#_Toc104499448)

[II daļa – Nepārraudzītā mašīnmācīšanās 11](#_Toc104499449)

[Hierarhiskā klasterizācija 11](#_Toc104499450)

[Hierarhiskās klasterizācijas algoritma hiperparametru apraksts 12](#_Toc104499451)

[K-vidējo algoritms 13](#_Toc104499452)

[K-vidējo algoritma hiperparametru apraksts 13](#_Toc104499453)

[II daļa – Secinājumi 13](#_Toc104499454)

[III daļa – Parraudzītā mašīnmācīšanās 14](#_Toc104499455)

[Naive Bayes algoritma pieejamie hiperparametri 14](#_Toc104499456)

[kNN algoritma pieejamie hiperparametri 14](#_Toc104499457)

[Neural Network algoritma pieejamie hiperparametri 14](#_Toc104499458)

[Informācija par testu un apmācību datu kopām 15](#_Toc104499459)

[3 veiktie train datu kopas eksperimenti kNN algoritmam 17](#_Toc104499460)

[3 veiktie train datu kopas eksperimenti Neural Network algoritmam 18](#_Toc104499461)

[Naive Bayes algoritma train datu kopas eksperiments 20](#_Toc104499462)

[Test datu kopas testēšana, izmantojot labākos hiperparametrus 20](#_Toc104499463)

[kNN algoritma test datu kopas eksperiments 20](#_Toc104499464)

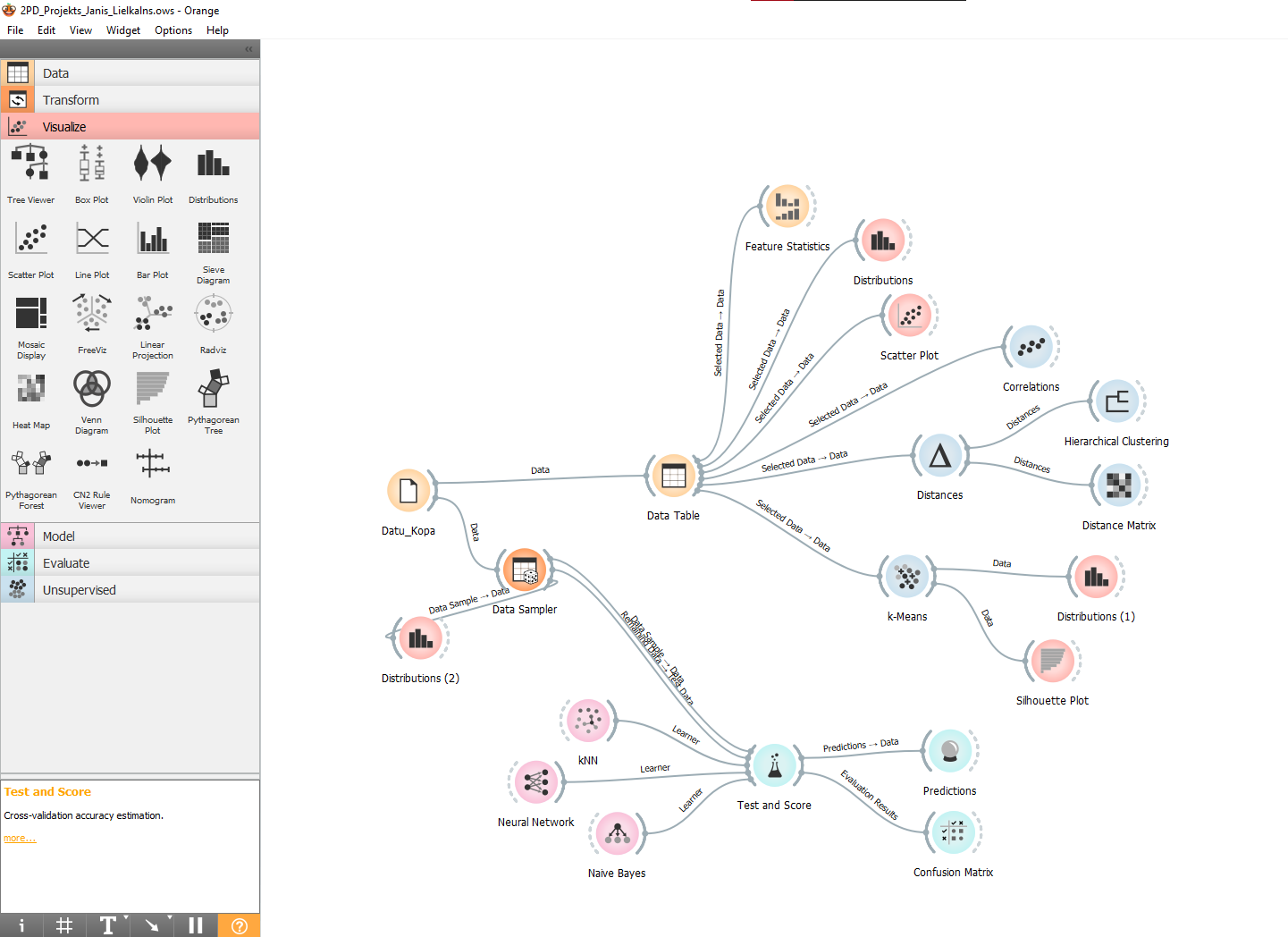
[Neural Network un Naive Bayes algoritmu test datu kopas eksperiments 20](#_Toc104499465)

[III daļa – Secinājumi 21](#_Toc104499466)

[Izmantotie informācijas avoti 22](#_Toc104499467)

Saite uz izveidoto projektu:

Saite uz izmantoto datu kopu: <https://archive.ics.uci.edu/ml/datasets/Raisin+Dataset>

 31.attēls

31. attēlā redzama kopējā, projektā iegūtā darbplūsma iekš Orange rīka.

# I Daļa– Datu pirmapstrāde/izpēte

### Datu kopas apraksts

Šajā projektā izmantota datu kopa “Raisin Dataset”.

Datu kopa un tās apraksts pieejams, izmantojot saiti: <https://archive.ics.uci.edu/ml/datasets/Raisin+Dataset>

Šis ir tiešais links uz datu kopas lejuplādi: <https://archive.ics.uci.edu/ml/machine-learning-databases/00617/>

Datu kopas autori: Ilkay Cinar, Murat Koklu, Sakir Tasdemir

Datu kopa tika izveidota 2020 gadā (atsauce: <https://dergipark.org.tr/tr/download/article-file/1227592>) un ir pieejama publiski kopš 2021. gada 1. aprīļa (2021-04-01) (open source licence) (atsauce: <https://archive.ics.uci.edu/ml/datasets/Raisin+Dataset>)

Šajā datu kopā tika izmantota iepriekš izstrādāta mašīnredzes sistēma, lai atšķirtu divas dažādas Turcijā audzētas rozīnes- Kecimen un Besni. Darbā tika apskatīti kopā 900 gabali rozīņu (450 gabali no katras šķirnes). Tālāk tika iegūti šo rozīņu attēli, kas tika pakļauti dažādiem pirmsapstrādes soļiem, kā arī tika veiktas 7 morfoloģisko pazīmju ekstrakcijas darbības. Šīs darbības tika veiktas, izmantojot attēlu apstrādes metodes. Katram objektam tika aprēķināta minimālā, vidējā, maksimālā un standartnovirzes statistiskā informācija. Abu rozīņu šķirņu dažādība un īpašības tika pārbaudītas un parādītas grafikos. Noslēdzot modeļi tika veidoti, izmantojot LR, MLP un SVM mašīnmācību tehniku, kā arī tika veikti veiktspējas mērījumi. (atsauce: <https://dergipark.org.tr/tr/download/article-file/1227592>)

### Datu kopas satura apraksts

Datu kopa sastāv no: 900 datu objektiem (450 objekti katrai klasei), 8 atribūtiem (1 no tiem klase- Class, kas norāda rozīnes šķirni). Šie atribūti parāda rozīnes īpašības: tās laukumu (**Area**) – norāda pikseļu daudzumu, ko aizņem rozīne; perimetru (**Perimeter**) – norāda vidi, aprēķinot attālumu starp rozīnes robežām jeb malām un pikseļiem ap to; galvenās ass garumu (**MajorAxisLenght**), tas ir, garāko līniju, kas var tikt novilkta uz rozīnes; sekundārās ass garumu (**MinorAxisLenght**) – īsākā līnija, kas var tikt novilkta uz rozīnes; ekscentriskums (**Eccentricity**) – dod lielumu no elipses ekscentriskuma, kurai ir tādi pat momenti kā rozīnēm; izliekts laukums (**ConvexArea**) – dod pikseļu daudzumu mazākajam izliekuma apvalkam no reģiona, ko veido rozīne; apjoms (**Extent**) – norāda ar rozīnēm veidotā apgabala attiecību pret kopējiem pikseļiem ierobežotajā rāmī. (atsauce: <https://dergipark.org.tr/tr/download/article-file/1227592>)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr. Pk.** | **Īpašība** | **Minimālā vērtība** | **Vidējā vērtība** | **Maksimālā vērtība** | **Standartnovirze** |
| 1. | Area (pikseļi) | 25387 | 87804.128 | 235047 | 39002.111 |
| 2. | Perimeter | 619.074 | 1165.907 | 2697.753 | 273.764 |
| 3. | MajorAxisLenght | 225.63 | 430.93 | 997.292 | 116.035 |
| 4. | MinorAxisLenght | 143.711 | 254.488 | 492.275 | 49.989 |
| 5. | Eccentricity | 0.349 | 0.782 | 0.962 | 0.09 |
| 6. | ConvexArea (pikseļi) | 26139 | 91186.09 | 278217 | 40769.29 |
| 7. | Extent | 0.38 | 0.7 | 0.835 | 0.053 |

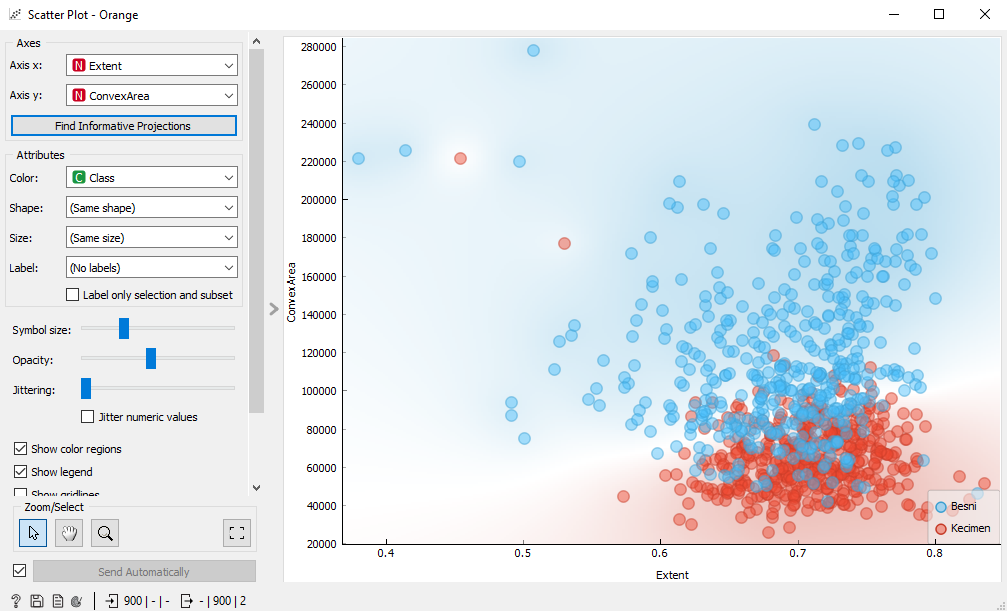
1.tabula

1. tabulā izmantoti dati, kas iegūti no 4. tabulas saitē: <https://dergipark.org.tr/tr/download/article-file/1227592>

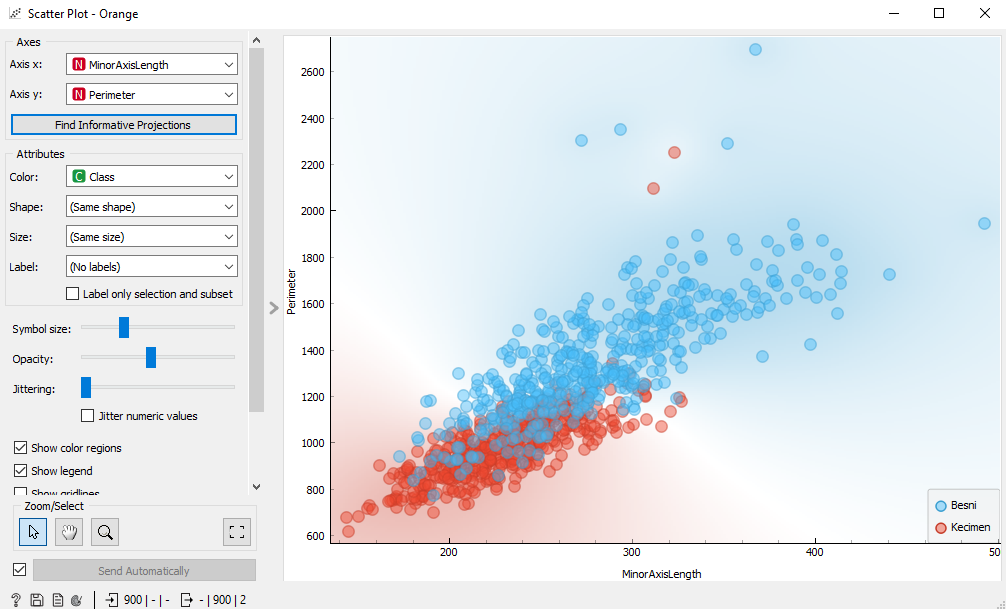
Attēls, kurā ir galds

Apraksts ģenerēts automātiski 1.attēls

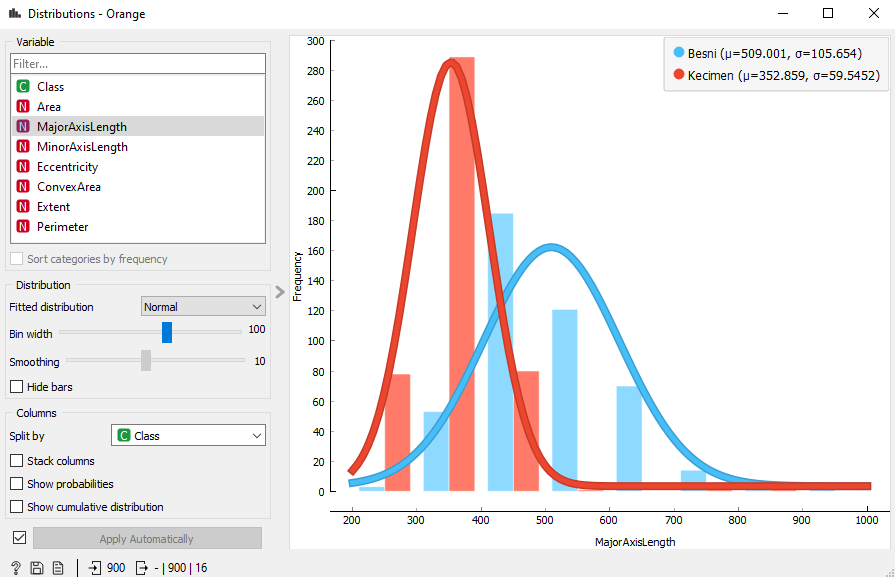
1. attēlā redzams datu faila struktūras fragments ar visām tā kolonnām un vērtībām pirmajiem 19 objektiem.

 2.attēls

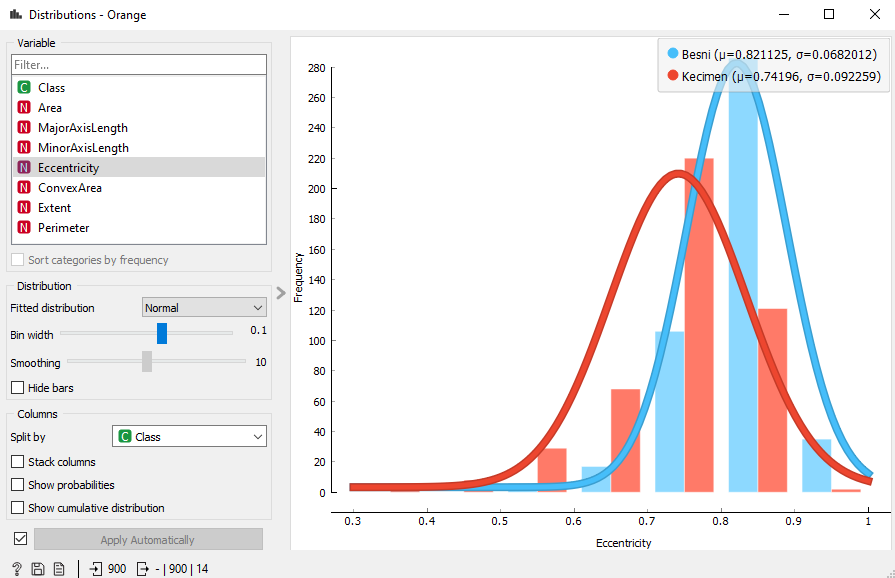
2. attēlā redzama pirmā no izveidotajām divdimensiju izkliedes diagrammām (**Scatter Plot**). Šajā gadījumā klases atdalāmība ilustrēta, izvēloties pazīmes **Extent** (x ass) un **ConvexArea** (y ass).

 3.attēls

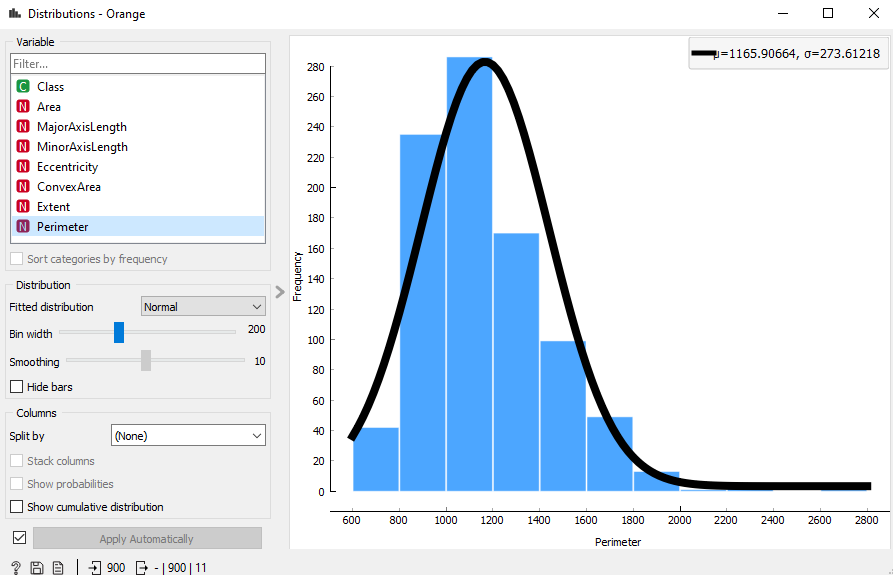
3. attēlā attēlota otrā izveidotā izkliedes diagramma. Šeit izvēlētas pazīmes **MinorAxisLenght** (x ass) un **Perimeter** (y ass).

 4.attēls

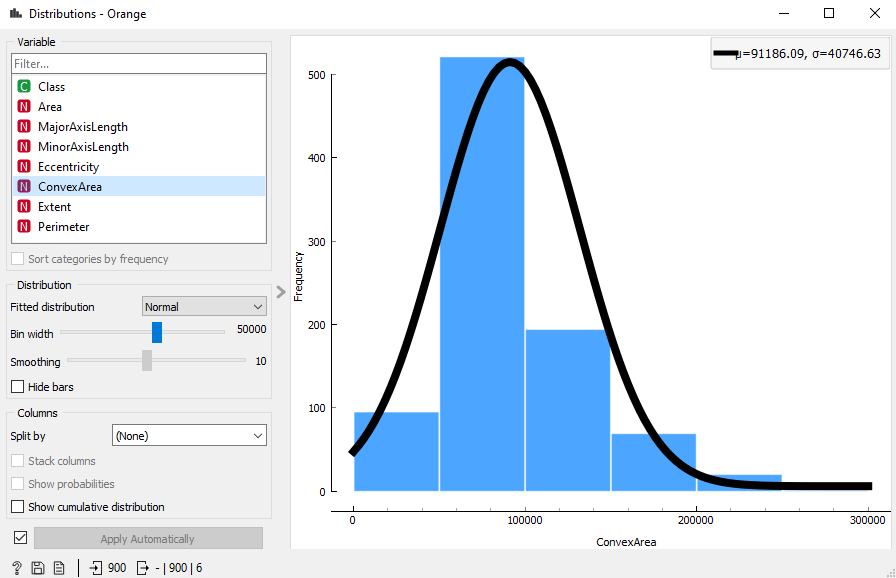
4. attēlā redzama pirmā izveidotā histogramma, kur klašu atdalīšanai izvēlēta pazīme **MajorAxisLenght.**

 5.attēls

5. attēls rāda histogrammu, kurā par pamatpazīmi izvēlēta **Eccentricity**.

 6.attēls

6. attēlā redzams atribūta **Perimeter** sadalījums.

 7.attēls

7. attēls parāda atribūta **ConvexArea** sadalījumu.

 8.attēls

8. attēlā redzamas visas statistiskās vērtības: ar vērtību **Mean** apzīmētas vidējās vērtības un **Dispersion**- dispersija jeb standartnovirze.

## I Daļa – Secinājumi

Pēc datu kopas apraksta, iespējams secināt, ka šīs datu kopas klases ir līdzsvarotas, jo katrai no tām pieder vienāds skaits objektu.

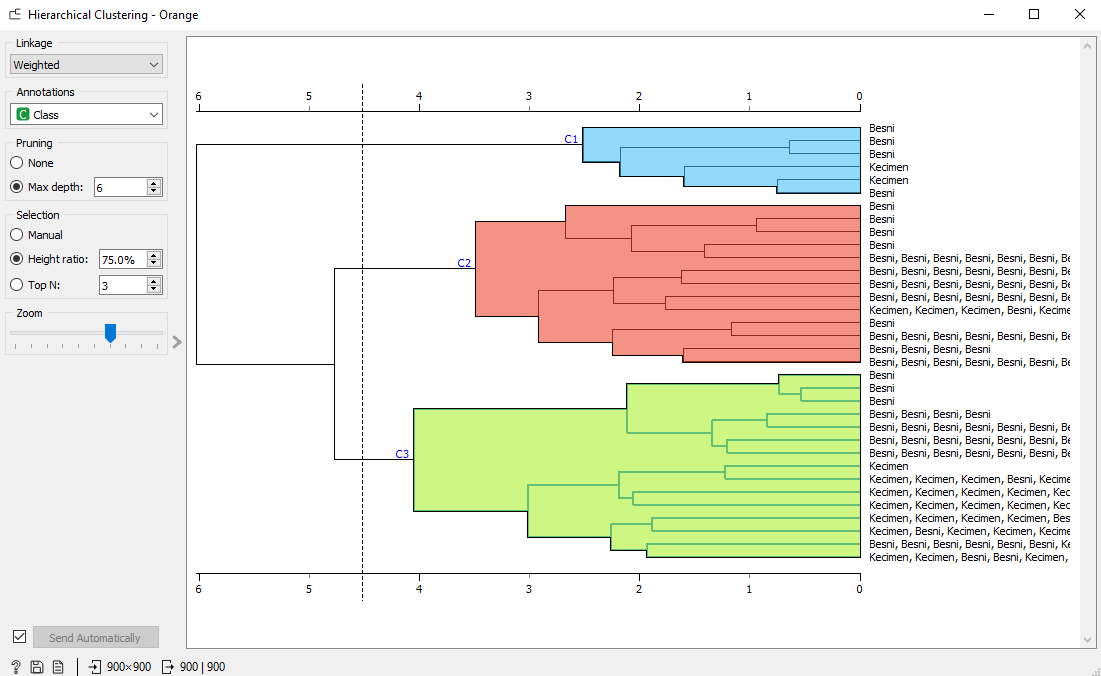
Apskatot izveidotās diagrammas, var apgalvot, ka datu vizuālais atspoguļojums ļauj redzēt datu struktūru, taču daļa objektu nav skaidri atdalāmi. Kā redzams konkrēti izveidotajās Scatter Plot diagrammās, sarkanā krāsā apzīmētā klase ir daudz ciešāk saistīta par zilo klasi, kas ir vairāk izkliedēta un nedaudz nobīdīta no sarkanās klases, tādēļ var secināt, ka atdalāmi datu grupējumi ir daļēji rodami.

Konkrēti šie datu grupējumi atrodas ļoti tuvu viens otram, un pat daļēji pārklājas viens ar otru.

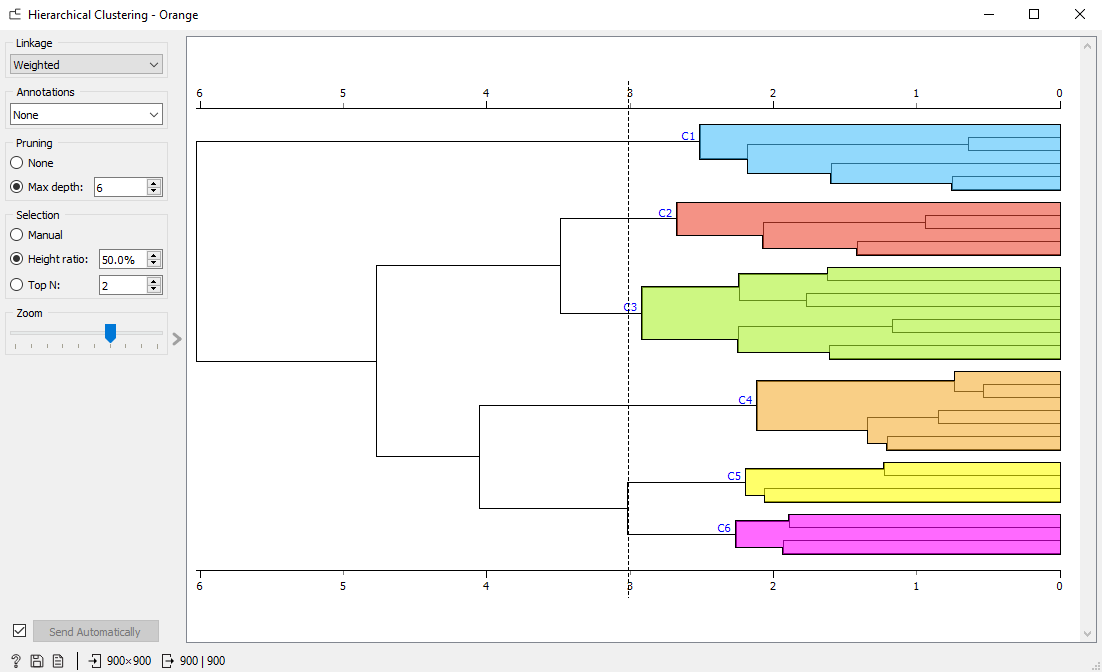
Pēc statistisko rādītāju analīzes var izteikt secinājumus, ka dispersija ir vislielākā pie īpašībām **ConvexArea** un **Area** (0.45 un 0.44) un vismazākā pie īpašībām **Eccentricity** un **Extent** (0.12 un 0.08). Līdzīgi ir ar maksimālajām un minimālajām vērtībām, kur vislielākā starpība starp tām ir īpašībām **ConvexArea** un **Area** (252078 un 209660), kur Max. un Min. vērtības atšķīrās par aptuveni 10 reizēm, un vismazākā starpība ir īpašībām **Eccentricity** un **Extent** (0.61 un 0.46), kur Max. un Min. vērtības atšķīrās par mazāk nekā 3 reizēm.

# II daļa – Nepārraudzītā mašīnmācīšanās

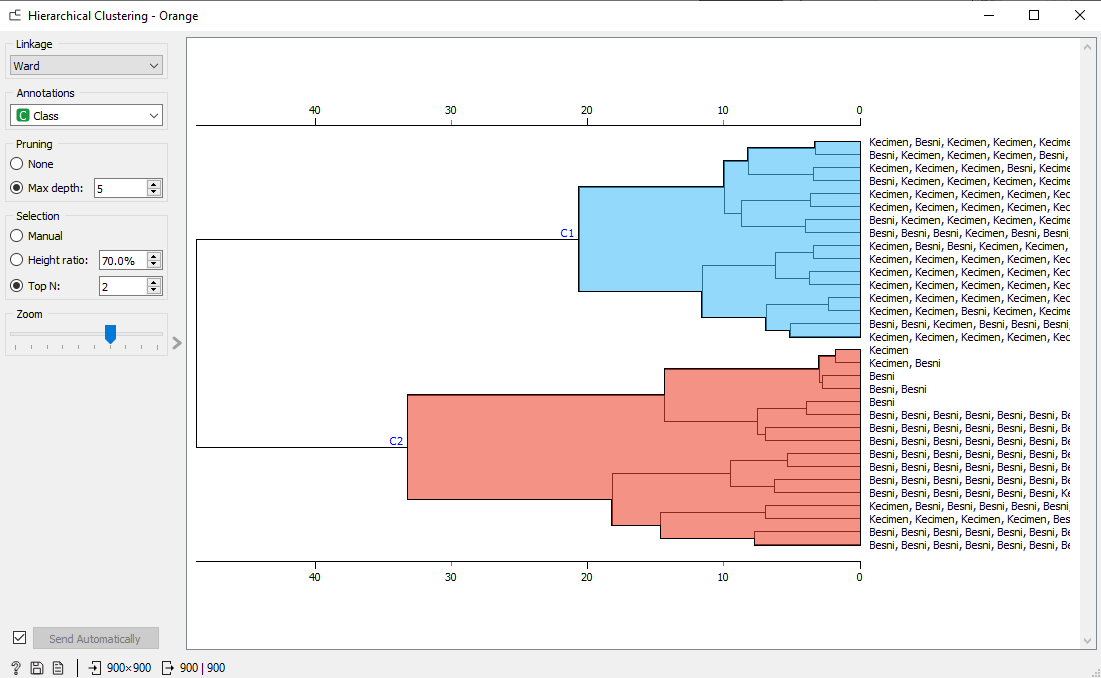
### Hierarhiskā klasterizācija

 9.attēls

9. attēlā redzamā rezultāta ieguvē tika izmantotas sekojošas hiperparametru vērtības: **Linkage**- ***Weighted***; **Pruning**- ***Max depth: 6***; **Selection**- ***Height ratio: 75%.*** Šeit redzami 3 klasteri- zilā, sarkanā un zaļā krāsā.

 10.attēls

10. attēla hierarhiskās klasterizācijas algoritma attēlošanai tika izmainīti šādi hiperparametri: **Annotations**- ***None***; **Selection**- ***Height ratio: 50%.*** Šajā gadījumā, samazinot ***Height ratio*** par 25%, tika iegūti 6 klasteri, kas iekrāsoti dažādās krāsās.

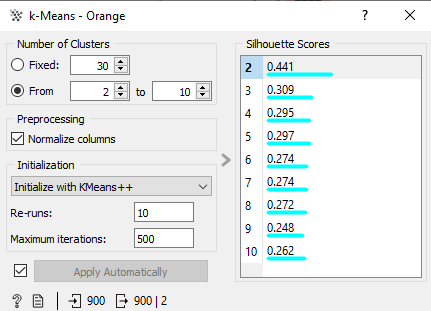
 11.attēls

11. attēla ieguvē tika mainītas sekojošas hiperparametru vērtības: **Linkage**- ***Ward***; **Pruning**- ***Max depth: 5***; **Selection**- ***Top N: 2.*** Te redzams, ka iegūti tikai divi klasteri, jo tā norādīts ar opcijas ***Top N*** palīdzību. Opcija ***Ward*** aprēķina kļūdas kvadrātu summas palielinājumu.

## Hierarhiskās klasterizācijas algoritma hiperparametru apraksts

Hierarhiskās klasterizācijas algoritmam Orange rīkā ir pieejami 5 hiperparametri, no tiem sekojošie 3 ietekmē dendrogrammu un tās vizuālo izskatu. **Linkage** hiperparametrs maina attāluma starp klasteriem mērīšanas veidus, kopā tādi ir 5: ***Single***; ***Average***; ***Weighted***; ***Complete***; ***Ward***. **Pruning** ļauj izvēlēties maksimālo dendrogrammas dziļumu, kas lielākoties maina tikai tās izskatu, ne pašu klasterēšanu. **Selection** piedāvā trīs opcijas: ***Manual***, kas ļauj manuāli atlasīt vienu vai vairākus klasterus iekš dendogrammas; ***Height ratio*** tiek automātiski izvēlēts, lietotājam noklikšķinot uz augšējās vai apakšējās mērjoslas, kas dendrogrammā ievieto vertikāli un atlasa visus dendrogrammas elementus pa labi no tās; ***Top N*** atlasa augšējo mezglu skaitu (klasteru skaitu). **Annotations** ļauj nomainīt dendrogrammas label (etiķeti). **Zoom** variē attēla, dendrogrammas tuvumu. (atsauce: <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/unsupervised/hierarchicalclustering.html>)

### K-vidējo algoritms

 12.attēls

12. attēlā redzams, ka izvēlētas ***k*** vērtības no 2 līdz 10, kā arī katra ***k*** **Silhouette Score** vērtības. Šīs **Silhouette Score** vērtības norāda klasteru atdalāmību, tas ir, jo šī vērtība ir tuvāk vieniniekam, jo labāka klasteru atdalāmība, un pretēji- jo tuvāk nullei, jo maznozīmīgāks ir dotais klasteru sadalījums. Konkrētajā gadījumā redzams, ka vislabākais sadalījums ir starp diviem klasteriem- **Silhouette Score: *0.441***, un ar katru papildus klasteru **Silhouette Score** samazinās, kas norāda uz sliktāku, nonozīmīgāku datu sadalījumu.

## K-vidējo algoritma hiperparametru apraksts

Kā redzams 11. attēlā, šim algoritmam ir 3 hiperparametri: **Number of Clusters**, **Preprocessing** un **Initialization**. **Number of Clusters** pieejamas divas opcijas: ***Fixed*** un ***From X to Y***. ***Fixed*** sagrupē datus līdz noteiktajam kopu skaitam, un ***From X to Y*** parāda atlasīto klasteru diapazona grupēšanas rezultātus (**Silhouette score**). **Preprocessing**, gadījumā ja šī opcija ir atlasīta, normalizē kolonnas, centrējot vidējo vērtību līdz nullei un mērogojot standartnovirzi uz vieninieku. **Initialization** ļauj izvēlēties, kā algoritms sāks klasterēšanu. ***Re- runs*** norāda cik reizes algoritms tiek veikts no nejaušām sākotnējām pozīcijām. ***Maximum iterations*** parāda algoritma izpildes maksimālo iterāciju skaitu. (atsauce: <https://orangedatamining.com/widget-catalog/unsupervised/kmeans/>)

## II daļa – Secinājumi

Analizējot abu algoritmu darbību, iespējams secināt, ka izmantotajā datu kopā esošās klases ir atdalāmas, taču ne ļoti labā līmenī, par ko norāda iegūtie Silhouette Score un hierarhiskajā klasterizācijā iegūtas dendrogrammas.

# III daļa – Parraudzītā mašīnmācīšanās

Šajā daļā tika brīvi izvēlēti **Naive Bayes** un **kNN** pārraudzītās mašīnmācīšanās algoritmi. **Naive Bayes** ir klasifikators no vienkāršu varbūtējo klasifikatoru saimes un tā pamatā ir ***Bayes*** teorēma, kas īsteno pieņēmumu par līdzekļu neatkarību. **Naive Bayes** algoritmu es izvēlējos tieši šī apraksta dēļ, jo mani ieinteresēja tas, ka šis algoritms raksturots kā vienkāršs klasifikators. **kNN** ir algoritms, kas meklē ***k*** tuvākos apmācības piemērus līdzekļu telpā un izmanto to vidējo vērtību kā paredzējumu (prediction). Šo algoritmu es izvēlējos, jo biju jau par to iepriekš dzirdējis, atšķirībā no pārējiem, tādēļ vēlējos to izmantot savā projektā.

(atsauces: <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/naivebayes.html> (Naive Bayes); <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/knn.html> (kNN))

### Naive Bayes algoritma pieejamie hiperparametri

Šim algoritmam Orange rīkā nav pieejamu hiperparametru, kas ietekmētu pašu algoritmu. Iespējams izmainīt tikai tā nosaukumu. (atsauce: <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/naivebayes.html>)

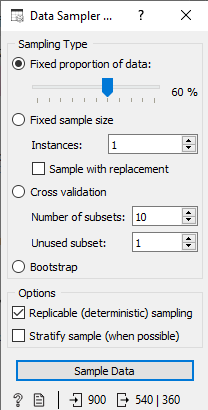
### kNN algoritma pieejamie hiperparametri

**kNN** algoritms piedāvā 3 hiperparametrus: **Neighbors**, **Metric**, **Weight**. **Neighbors**- ***Number of neighbors*** ļauj iestatīt tuvāko kaimiņu skaitu, **Metric** maina attāluma parametrus. Tas ļauj izvēlēties starp ***Euclidean*** (tas ir attālums starp diviem punktiem taisnā līnijā); ***Manhattan*** (visu atribūtu absolūto atšķirību summa); ***Mahalnobis*** (attālums starp punktu un izkliedi). **Weight** dod izvēli starp ***Uniform*** (visi punkti katrā apgabalā tiek svērti vienādi) vai ***Distance*** (vaicājuma punkta tuvākajiem kaimiņiem ir lielāka ietekme nekā tālākiem kaimiņiem). (atsauce: <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/knn.html>)

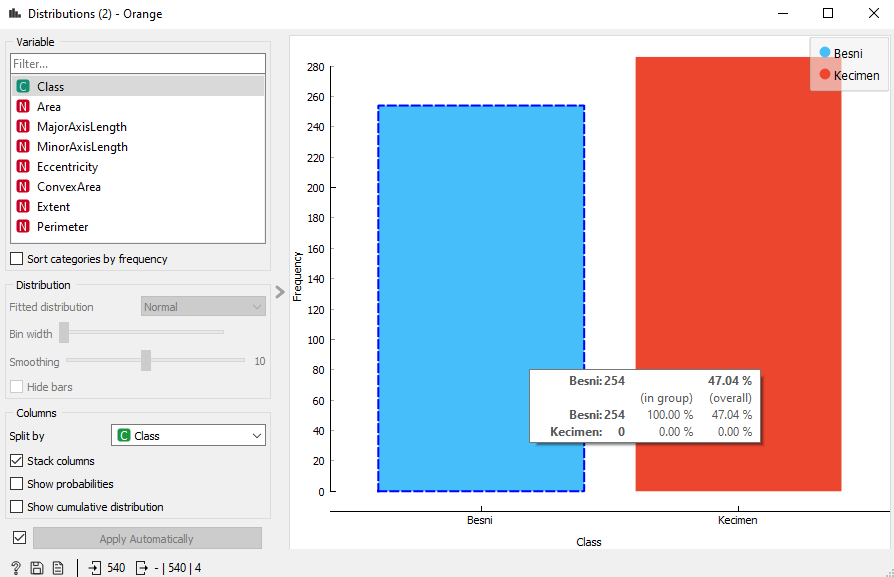
### Neural Network algoritma pieejamie hiperparametri

Neural Network algoritmam pieejami 5 parametri: Neurons per hidden layer- apzīmē neironu skaitu slēptajā slānī; Activation dotas četras izvēles: Identity (bezplūsmas aktivizēšana), Logistic (loģistiskā sigmoīda funkcija), tanh (hiperboliskā tangenssa funkcija), ReLu (rektificētas lineārās vienības funkcija); Solver: L-BFGS-B (optimizētājs kvazi-Ņūtonā metožu saimē), SGD (stohastiska gradienta nolaišanās), Adam (stohastisks gradienta optimizētājs); Alpha- L2 soda parametrs; Max iterations- maksimālais iterāciju skaits. (atsauce: <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/neuralnetwork.html>)

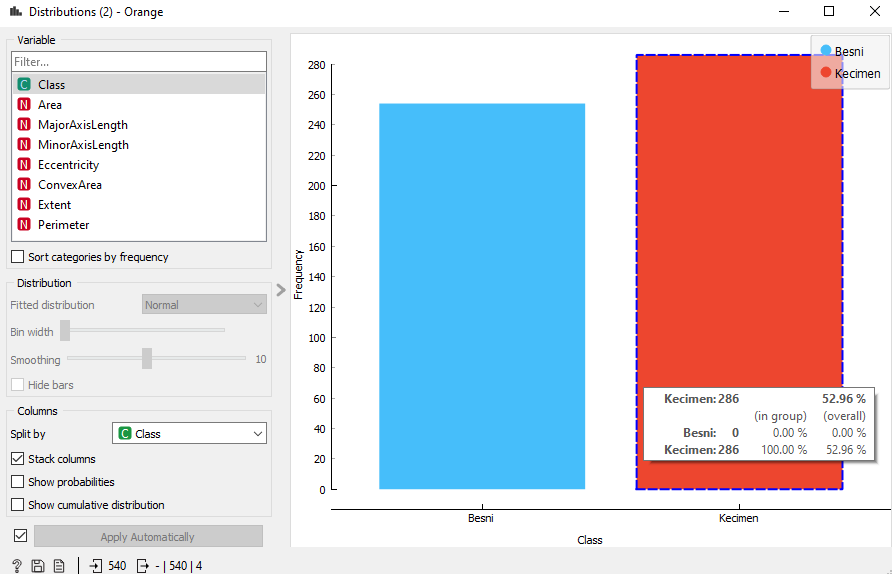
### Informācija par testu un apmācību datu kopām

 13.attēls

13. attēlā redzams testa un apmācību datu kopām pievienoto datu objektu skaits. Šeit: 60% jeb 540 objekti.

 14.attēls

14. attēlā redzams Besni klases datu objektu skaits, kas iekļauts apmācību un testa datu kopās: 254 jeb 47.04%.

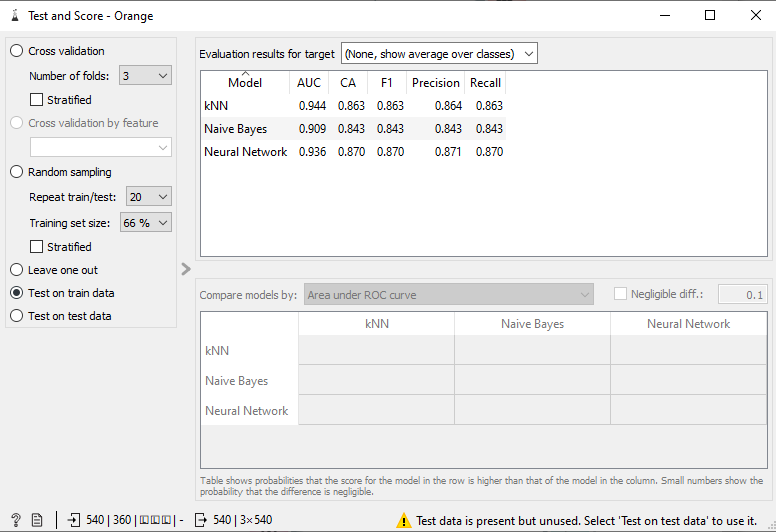
 15.attēls

15. attēlā redzams Kecimen klases datu objektu skaits, kas iekļauts apmācību un testa datu kopās: 286 jeb 52.96%.

### 3 veiktie train datu kopas eksperimenti kNN algoritmam

Attēls, kurā ir teksts

Apraksts ģenerēts automātiski 16.attēls



17.attēls

16. un 17. attēlā redzami pirmā veiktā eksperimenta hiperparametri un rezultāti kNN algoritmam.

Attēls, kurā ir teksts

Apraksts ģenerēts automātiski 18.attēls Attēls, kurā ir galds

Apraksts ģenerēts automātiski 19.attēls

18., 19. attēlā doti otrā eksperimenta hiperparametri un rezultāti.

Attēls, kurā ir teksts

Apraksts ģenerēts automātiski 20.attēls Attēls, kurā ir galds

Apraksts ģenerēts automātiski 21.attēls

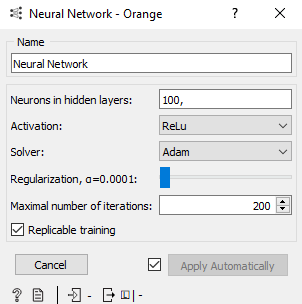
20., 21. attēlā attēloti trešā eksperimenta hiperparametri un rezultāti.

|  |  |  |  |
| --- | --- | --- | --- |
| **Neighbors** | **1.eksperiments** | **2.eksperiments** | **3.eksperiments** |
| Number of neighbors | 5 | 3 | 8 |
| Metric | Euclidean | Manhattan | Chebyshev |
| Weight | Uniform | Uniform | Uniform |

2.tabula

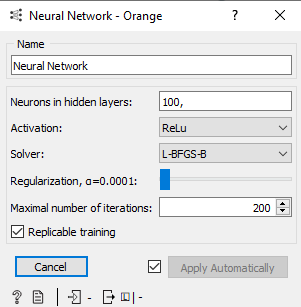
2. tabulā atspoguļoti **kNN** algoritma train eksperimentos izmantotās hiperparametru vērtības. Ar dzeltenu apzīmēts eksperiments, kura izvēlētie hiperparametri nodrošināja vislabāko algoritma veiktspēju.

### 3 veiktie train datu kopas eksperimenti Neural Network algoritmam

 22.attēls Attēls, kurā ir galds

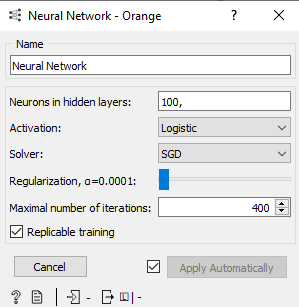
Apraksts ģenerēts automātiski 23.attēls

22., 23. attēlā redzami pirmā Neural Network algoritma eksperimenta hiperparametri un rezultāti.

 24.attēls Attēls, kurā ir galds

Apraksts ģenerēts automātiski 25.attēls

24., 25. attēlā attēloti otrā eksperimenta hiperparametri un rezultāti.

 26.attēls Attēls, kurā ir galds

Apraksts ģenerēts automātiski 27.attēls

26., 27. attēls parāda trešā eksperimenta hiperparametrus un rezultātus.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hiperparametri** | **1.eksperiments** | **2.eksperiments** | **3.eksperiments** |
| Neurons in hidden layers | 100, | 100, | 100, |
| Activation | ReLu | ReLu | Logistic |
| Solver | Adam | L-BFGS-B | SGD |
| Regularization, **α** | 0.0001 | 0.0001 | 0.0001 |
| Maximal number of iterations | 200 | 200 | 400 |

3.tabula

3. tabulā atspoguļoti **Neural Network** algoritma train eksperimentos izmantotās hiperparametru vērtības. Ar dzeltenu apzīmēts eksperiments, kura izvēlētie hiperparametri nodrošināja vislabāko algoritma veiktspēju.

### Naive Bayes algoritma train datu kopas eksperiments

Attēls, kurā ir galds

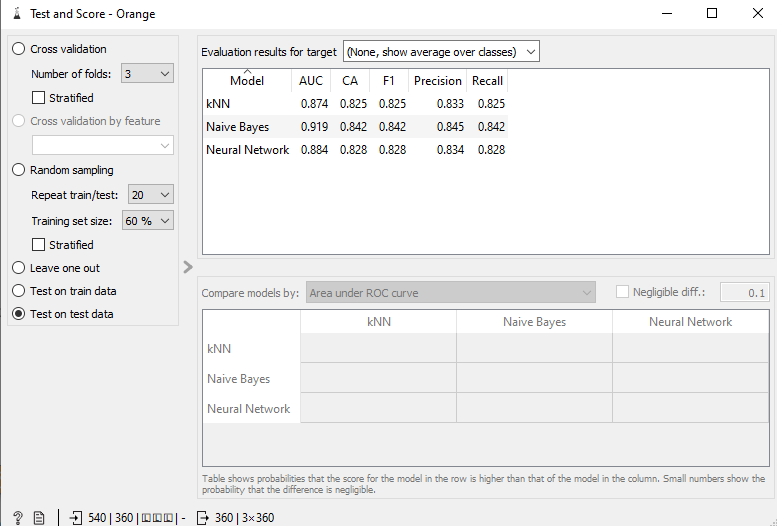
Apraksts ģenerēts automātiski 28.attēls

28. attēlā redzamas **Naive Bayes** train eksperimentā iegūtās vērtības. Šajā gadījumā nav norādīti hiperparametri, jo tādu šim algoritmam nav, līdz ar to rezultāts ir nemainīgs.

## Test datu kopas testēšana, izmantojot labākos hiperparametrus

### kNN algoritma test datu kopas eksperiments

Šajā testā tiks izmantoti **kNN** train datu kopas 2. eksperimentā iegūtie hiperparametri.

 29.attēls

29. attēlā redzamas **kNN** algoritma test datu kopas eksperimentā iegūtie rezultāti.

### Neural Network un Naive Bayes algoritmu test datu kopas eksperiments

Šajā testā tiks izmantoti **Neural Network** train datu kopas 2. eksperimentā iegūtie hiperparametri.

Attēls, kurā ir galds

Apraksts ģenerēts automātiski 30. attēls

30. attēlā atspoguļotas **Neural Network** un **Naive Bayes** algoritmu test datu kopas eksperimentā iegūtie rezultāti.

## III daļa – Secinājumi

Apskatot train testos iegūtos rezultātus, var neapšaubāmi secināt, ka vislabākā algoritma veiktspēja no visu algoritmu eksperimentiem, tika prognozēta **Neural Network** algoritmam, ar 2. eksperimentā izmantotajiem hiperparametriem. Otrā labākā prognoze tika saņemta **kNN** algoritmam. Eksperimentējot ar **Neural Network** parametriem, train datu kopas vislabākā algoritma veiktspēja tika paredzēta izvēloties tieši ***L-BFGS-B*** kā **Solver** opciju. Interesanti ir tas, ka tika prognozēta gandrīz vai ideāla algoritma veiktspēja, taču, izmantojot katra algoritma labākos, eksperimentos atklātos hiperparametrus, atklājās, ka **Neural Network** šajā gadījumā tomēr nenodrošina vislabāko algoritma veiktspēju. Noslēdzot izrādījās, ka vislabākā veiktspēja ir tieši “vienkāršajam” **Naive Bayes** algoritmam, neņemot vērā faktoru, ka tam nav maināmu hiperparametru. Vissliktākā veiktspēja atklājās **kNN** algoritmam, kas train datu kopas eksperimentos uzrādīja otro labāko prognozi. Autorprāt, principāli šis parāda, ka **Naive Bayes** algoritms ir ar vislabāko veiktspēju, lai gan ir grūti to apstiprināt, jo, lai būtu pilnībā pārliecināts par šī apgalvojuma patiesumu, būtu noteikti jāviec vēl papildus eksperimentu ar izvēlētajiem algoritmiem.

# Izmantotie informācijas avoti

1. <https://archive.ics.uci.edu/ml/datasets/Raisin+Dataset>
2. <https://archive.ics.uci.edu/ml/machine-learning-databases/00617/>
3. <https://dergipark.org.tr/tr/download/article-file/1227592>
4. <https://www.youtube.com/watch?v=dKURyzjh5Gc&t=1811s>
5. <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/unsupervised/hierarchicalclustering.html>
6. <https://orangedatamining.com/widget-catalog/unsupervised/kmeans/>
7. <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/naivebayes.html>
8. <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/knn.html>
9. <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/neuralnetwork.html>