

RĪGAS TEHNISKĀ UNIVERSITĀTE

Datorzinātnes un informācijas tehnoloģijas fakultāte

2.praktiskais darbs

mācību priekšmetā

“Mākslīgā intelekta pamati”

**Mašīnmācīšanās algoritmu lietojums**

Izstrādāja: Ralfs Vasils 201RDB375

Pārbaudīja: lektors Asoc.prof. A Anohina-Naumeca

Saite uz izveidoto projektu un datu kopu: [**https://github.com/raflsrafls/2studdarbs**](https://github.com/raflsrafls/2studdarbs)

2021./22. māc. gads

Saturs

Saturs

[**1.** **Datu pirmapstrāde/izpēte** 3](#_Toc104456572)

[**1.1 Datu kopas apraksts** 3](#_Toc104456573)

[**1.2 Datu kopas satura apraksts** 4](#_Toc104456574)

[**1.3** **Secinājumi par novērojumiem** 9](#_Toc104456575)

[**2.** **Nepārraudzītā mašīnmācīšanās** 9](#_Toc104456576)

[**2.1 Hierarhiskā klasterizācija** 9](#_Toc104456577)

[**2.2 K-vidējo algoritms** 13](#_Toc104456578)

[**3.** **Pārraudzītā mašīnmācīšanās** 16](#_Toc104456579)

[**3.1 Hiperparametru apraksts** 16](#_Toc104456580)

# **Datu pirmapstrāde/izpēte**

## **1.1 Datu kopas apraksts**

Datu kopa bija ņemta no tīmekļa vietnes “Kaggle”, <https://www.kaggle.com/datasets/muratkokludataset/rice-dataset-commeo-and-osmancik>.

Aprakstā autors ir norādījis saiti uz kuru doties, lai nolādētu datu kopu - <https://www.muratkoklu.com/datasets/> . Dodoties uz mājaslapu tika iegūta aprakstā norādītā datu kopa “Rice Dataset Cammeo and Osmancik”.

Izvēlētā datu kopa ir par datiem, kas salīdzina divas rīsu šķirnes Cammeo un Osmancik. Darba autors ir norādijis citēšanas pieprasījumu, tāpēc tiek iekļauti (licencēšanas nosacījumi) –

CINAR, I. and KOKLU, M., (2019). “Classification of Rice Varieties Using Artificial Intelligence Methods.” International Journal of Intelligent Systems and Applications in Engineering, 7(3), 188-194.

DOI: <https://doi.org/10.18201/ijisae.2019355381>

UCI Machine Learning Repository: <https://archive.ics.uci.edu/ml/datasets/Rice+%28Cammeo+and+Osmancik%29>

Šīs datu kopas autors ir “Murat Koklu”, kurš to izstrādājis 2019.gadā. Datu kopa ir radīta, lai mašīnmācības ceļā dators būtu spējīgs klasificēt rīsus no divām šķirnēm. Dati priekš datu kopu tika savākti izmantojot 3810 rīsu gabalu attēlus un no tiem tika iegūtas 7 morfoloģiskās pazīmes, kas tiek attēlotas datos.

Oriģinālā datu kopa bija xlsm formātā, bet to pārveidoju uz csv, lai tas atbilstu nosacījumiem, bez datu zaudēšanas.

## **1.2 Datu kopas satura apraksts**

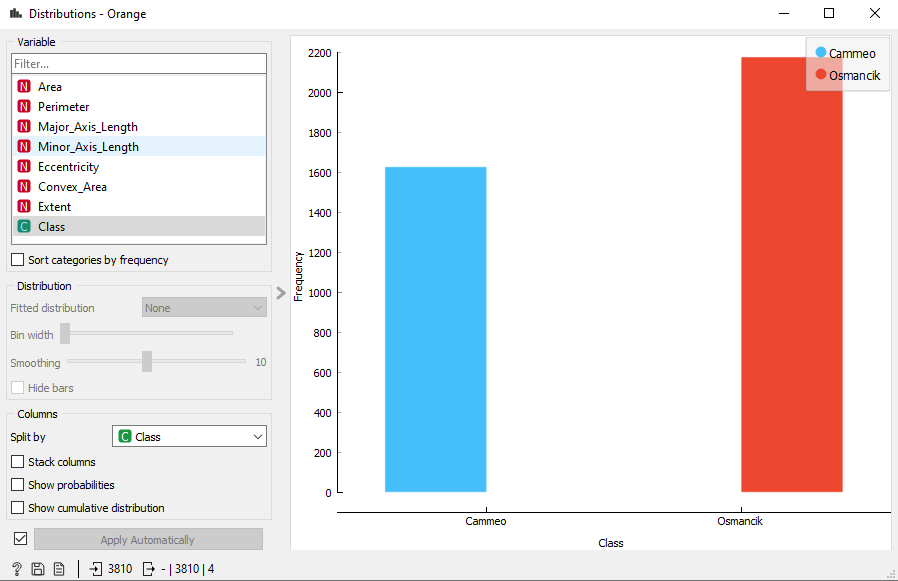
Izmantojot Orange programmatūras funkciju “Data table”, ir iespējams precīzi apskatīties cik datu kopā ir datu objektu. Šajā gadījumā ir 3810 datu objekti un nav trūkstošu datu (Attēls 1.1), kopā ir 8 atribūti. (Attēls 1.3)

Attēls, kurā ir galds

Apraksts ģenerēts automātiski

Attēls 1.1

Datu kopā ir 2 klases – “Cammeo” un “Osmancik”, klasei “Cammeo” ir 1630 atribūti un “Osmancik” klasei ir 2180 atribūti: (Attēls 1.2)



Attēls 1.2

Attēls, kurā ir teksts, iekštelpa, ekrānuzņē​​​mums

Apraksts ģenerēts automātiski

Attēls 1.3

Kopējais atribūtu skaits , vērtību tipi un diapazoni, (Attēls 1.4) (Tabula 1.1)



Attēls 1.4

Tabula 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| n.p.k | Atribūta apzīmējums | Min | Max |
| 1 | Area | 7551 | 18913 |
| 2 | Perimeter | 359.1 | 548.446 |
| 3 | Major\_Axis\_Length | 145.264 | 239.01 |
| 4 | Minor\_Axis\_Length | 59.5324 | 107.542 |
| 5 | Eccentricity | 0.777233 | 0.948007 |
| 6 | Convex\_Area | 7723 | 19099 |
| 7 | Extent | 0.497413 | 0.86105 |

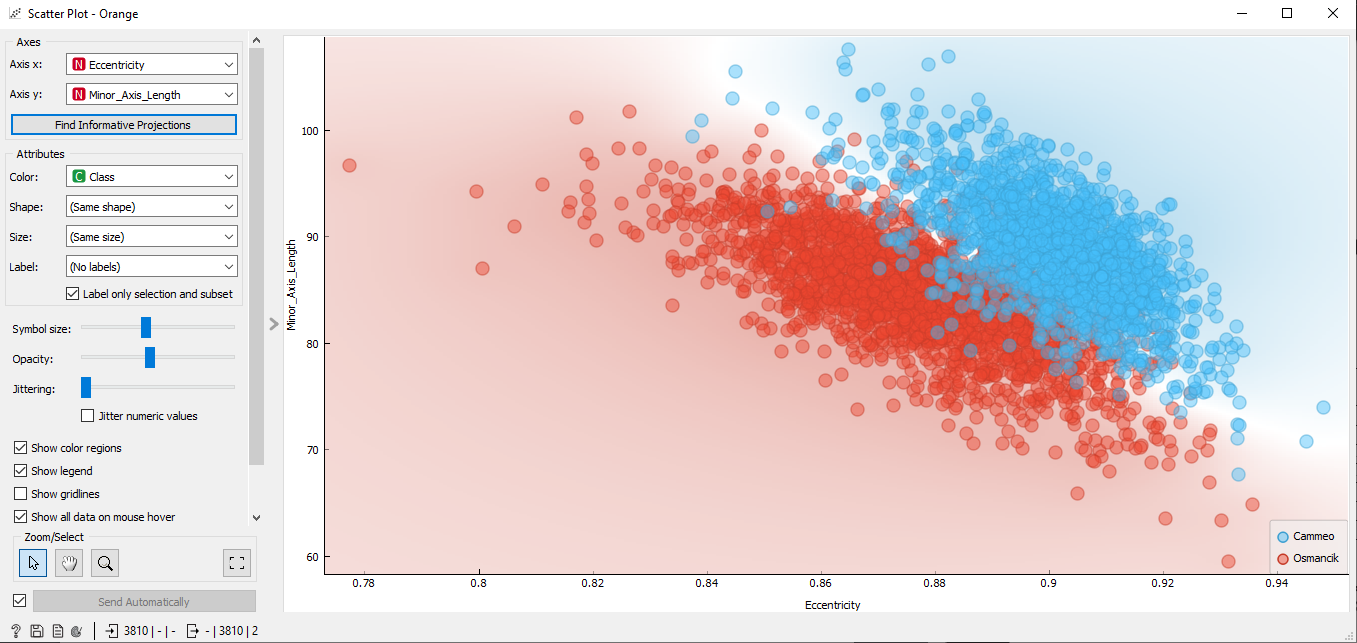
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n.p.k | Atribūta apzīmējums | Vidējās vērtības | Mediāna | Dispersija |
| 1 | Area | 12667.73 | 12421.50 | 0.14 |
| 2 | Perimeter | 454.239 | 448.852 | 0.0783561 |
| 3 | Major\_Axis\_Length | 188.776 | 185.81 | 0.0924184 |
| 4 | Minor\_Axis\_Length | 86.3138 | 86.4346 | 0.0663749 |
| 5 | Eccentricity | 0.886871 | 0.88905 | 0.0234701 |
| 6 | Convex\_Area | 12952.50 | 12706.50 | 0.14 |
| 7 | Extent | 0.661934 | 0.645361 | 0.116671 |

Tabula 1.2 (Statistiskie rādītāji)

Izmantojot izkliedes diagrammas jeb “Scatter plot”, tiek ilustrētas klases atdalamības atribūtiem. Šajā gadījumā atribūti Major\_Axis\_Length + Extent (Attēls 1.5) un Eccentricity + Minor\_Axis\_Length (Attēls 1.6).



Attēls 1.5

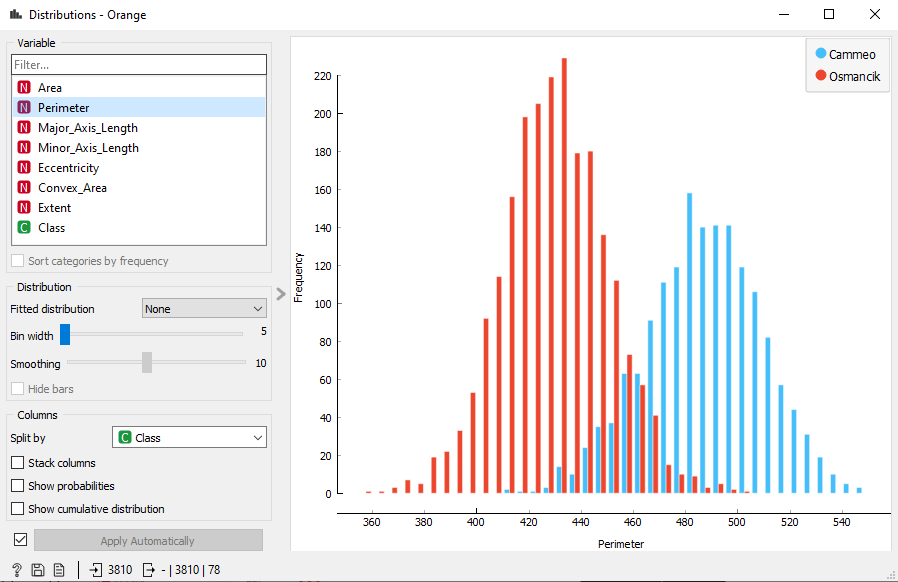


Attēls 1.6

Šajā gadījumā (Attēls 1.5) redzams, ka Osmancik rīsu šķirnes galvenās ass garums ir mazāks un apjoms līdzīgs salīdzinot ar Cammeno šķirni.

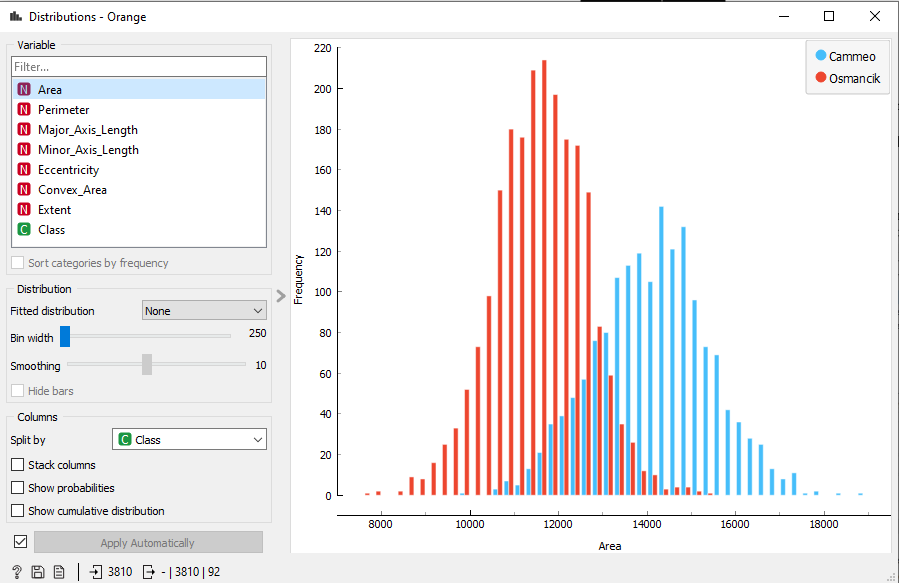
Un (Attēls 1.6) redzams, ka mazās ass garums ir līdzīgs un ekscentrisms ir lielāks Cammeo šķirnei.

Histogrammā (Attēls 1.7 un Attēls 1.10), attēlots, ka Osmancik šķirnes rīsiem ir lielāks perimetrs(Perimeter)

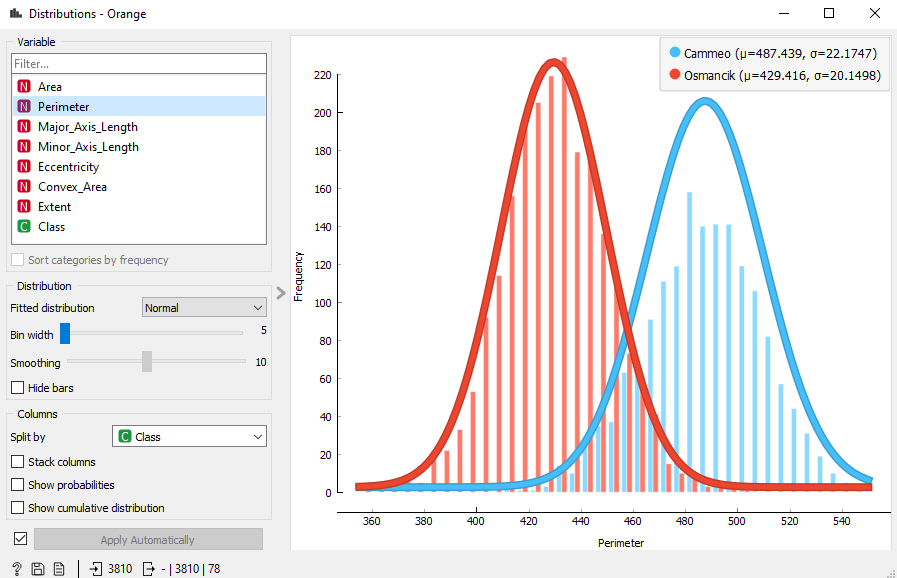


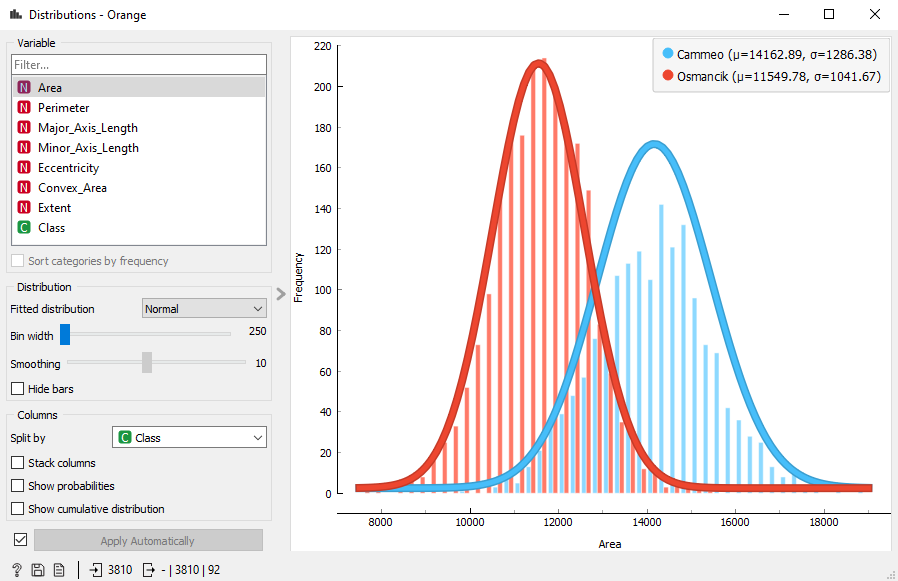
Attēls 1.7

Histogrammā (Attēls 1.8 un Attēls 1.9), attēlots, ka Osmancik rīsiem ir arī lielāka platība(Area)



Attēls 1.8





Attēls 1.10

Attēls 1.9

## **Secinājumi par novērojumiem**

Balstoties uz novērojumiem un izpētot datus par abām rīsu šķirnēm, ir iespējams noteikt, ka Osmancik šķirnes rīsi lielākoties dominē, bet ne par daudz, ir iespējams novērot šo secinājumu vislabāk Distributions rīkā(Attēls 1.2 un Attēls 1.4).

Vizuālais datu atspoguļojums konkrēti attēlo, ka ir viegli attdalīt abas klases un izpētīt datu struktūru(Attēls 1.5 un Attēls 1.6).

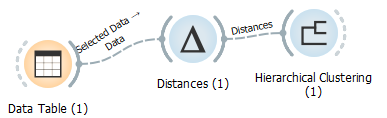
Datu grupējumi atrodas tuvu viens otram, to ir iespējams labāk izsecināt izpētot histogrammas(Attēls 1.7 un Attēls 1.8)

# **Nepārraudzītā mašīnmācīšanās**

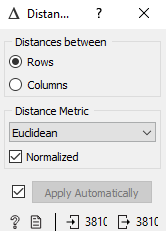
## **2.1 Hierarhiskā klasterizācija**

Lai veiktu hierarhisko klasterizāciju sākumā Datu Tabulas(Data Table) tiek pievienots logrīks “Distances”. Klāt pie “Distances” pievieno logrīku “Hierarchical clustering”(Attēls 2.1). Distances logrīks aprēķinās “Euclidean” attālumus starp datiem(Attēls 2.2).

Attēls 2.1



Attēls 2.2



**“Distances” hiperparametri un to nozīme(Attēls 2.2) –**

**Distances between –**

“Rows” vai “Columns” – izvēle vai mērīt attālumus starp rindām vai kolonnām,

**Distance Metric –**

Jāizvēlas piemērotā attāluma metrika:

“Euclidean” – Attālums starp diviem punktiem.

“Manhattan” – Absolūto atšķirību summa visiem atribūtiem.

“Cosine” – Leņķa kosinuss starp diviem iekšējās produktu telpas vektoriem.

“Jaccard” – Krustojuma lielums dalīts ar paraugu kopu savienības lielumu.

“Spearman” – Lineāra korelācija starp vērtību rangu, pārzīmēta kā attālums [0, 1] intervālā.

“Spearman absolute” - Lineāra korelācija starp absolūto vērtību rangu, pārzīmēta kā attālums [0, 1] intervālā.

“Pearson” - lineāra korelācija starp vērtībām, pārzīmēta kā attālums [0, 1] intervālā.

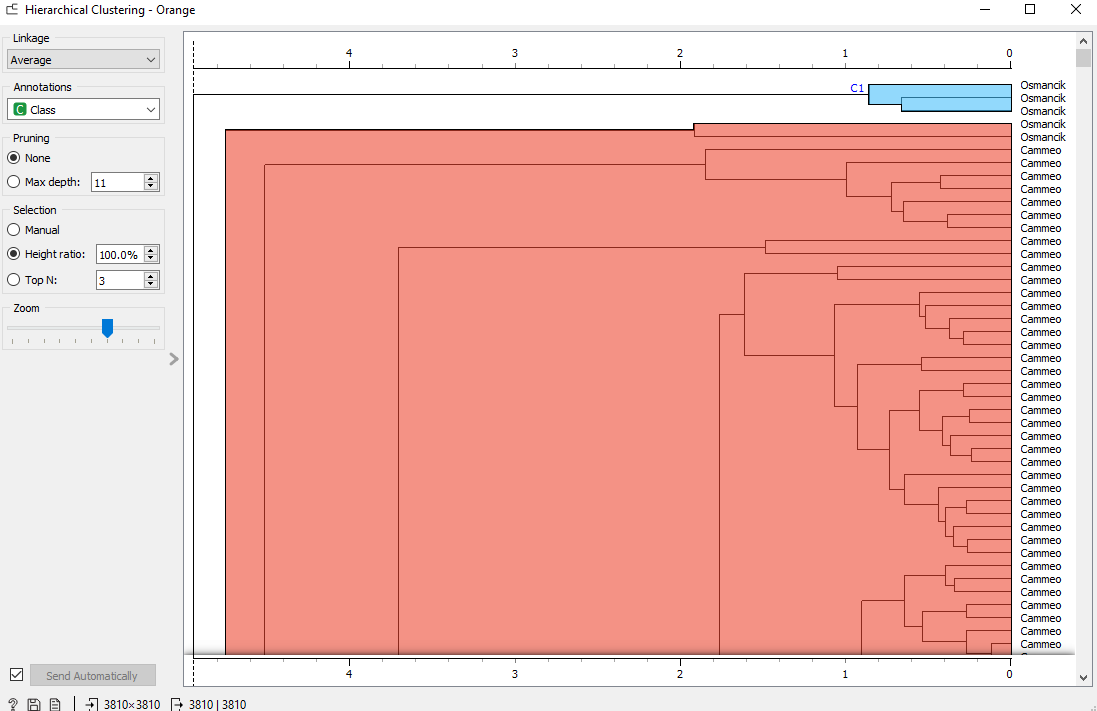
“Pearson absolute” - lineāra korelācija starp absolūtajām vērtībām, pārzīmēta kā attālums [0, 1] intervālā.

“Hamming” - pazīmju skaits, kurām atbilstošās vērtības atšķiras.

“Bhattacharyya distance” - līdzība starp diviem varbūtības sadalījumiem, nevis reāls attālums, jo tas neatbilst trīsstūra nevienlīdzībai

**ATSAUCE - https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/unsupervised/distances.html**

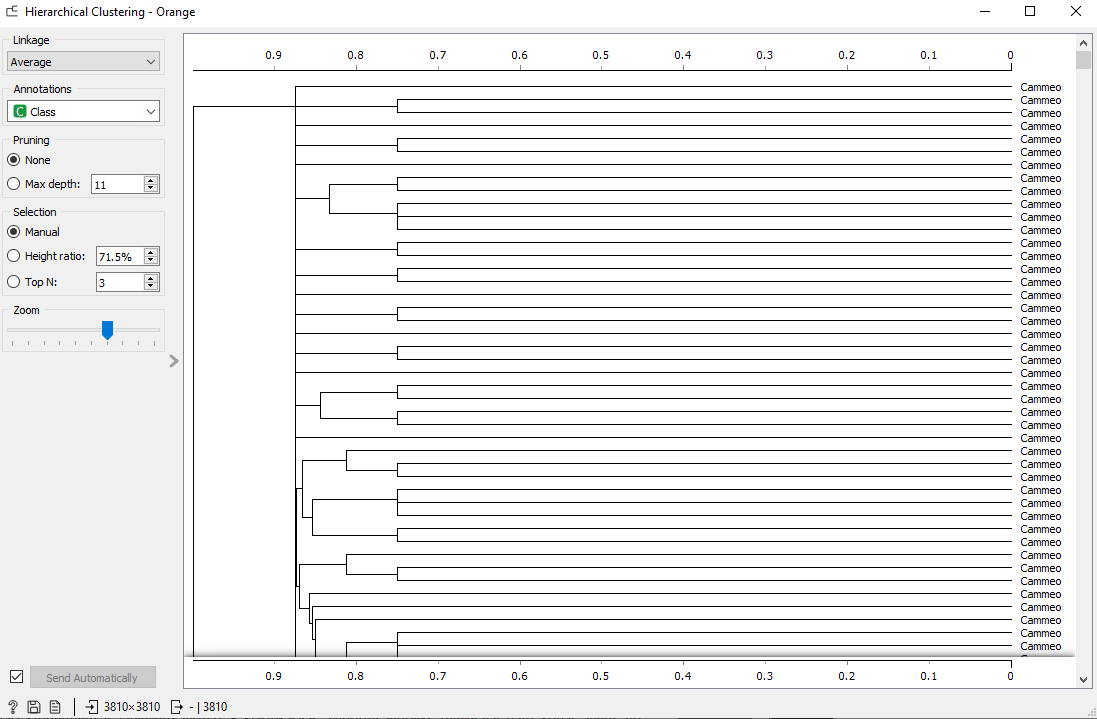
Atverot logrīku “Hierarchical Clustering”, Linkage tiek izvēlēts “Average” un Pruning “None”, un “Annotations” būs “Class”.(Attēls 2.3)



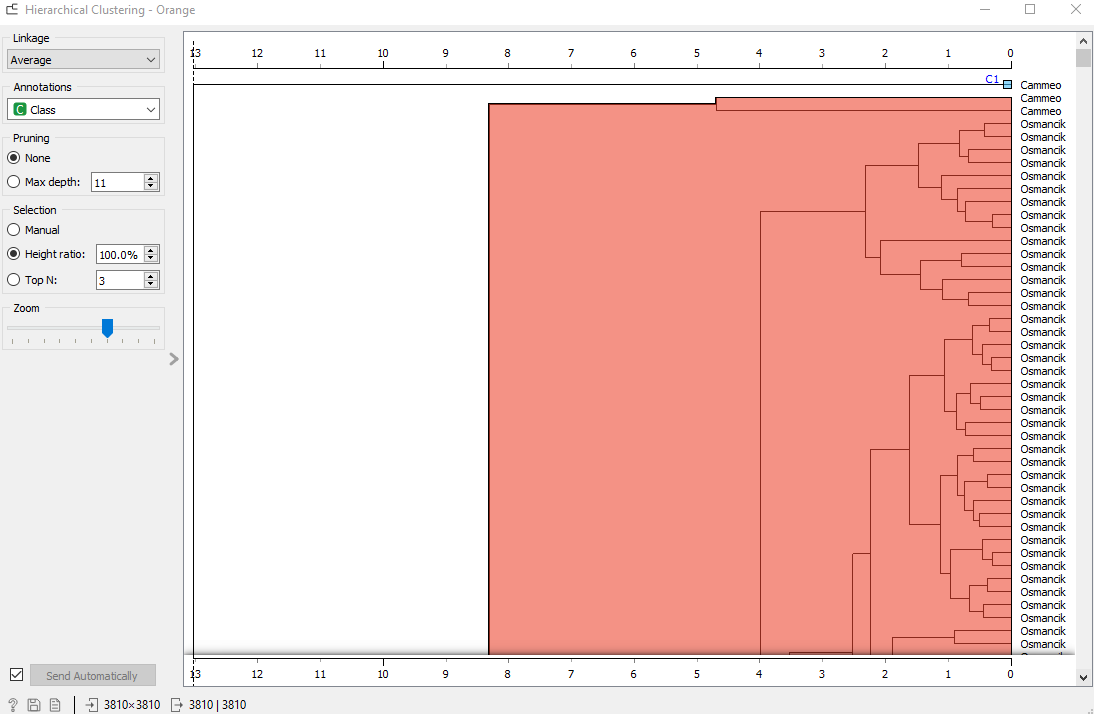
Attēls 2.3

Pēc iegūtajiem rezultātiem tiek izmainīti “Distance Metric”, kas iepriekš bija Euclidean, tagad tiek izmainīts uz “Hamming”, veicot šīs izmaiņas ir redzams, ka dati ir daudz labāk atdalīdi un vieglāk pārskatāmi.(Attēls 2.4)

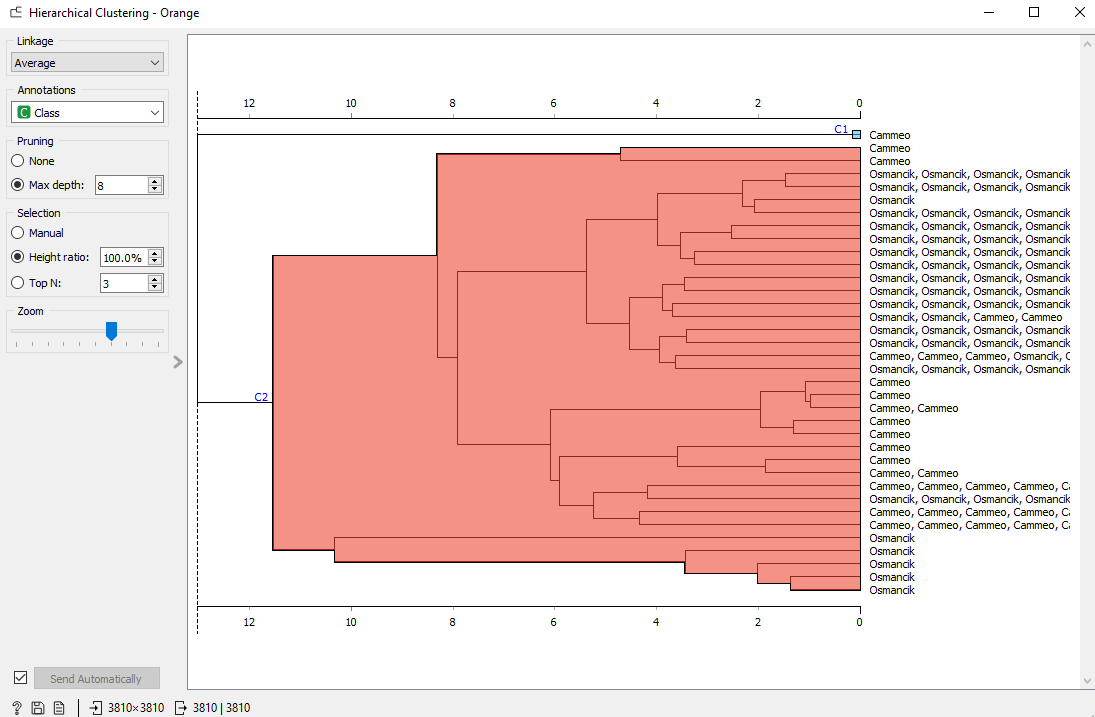
Atkārtoti nomainot “Distance Metric”, šoreiz uz “Manhattan” vēl vieglāk pārskatamus datus un to sadalījumu(Attēls 2.5) salīdzinot ar iepriekš iegūtajiem rezultātiem.

 Attēls 2.4

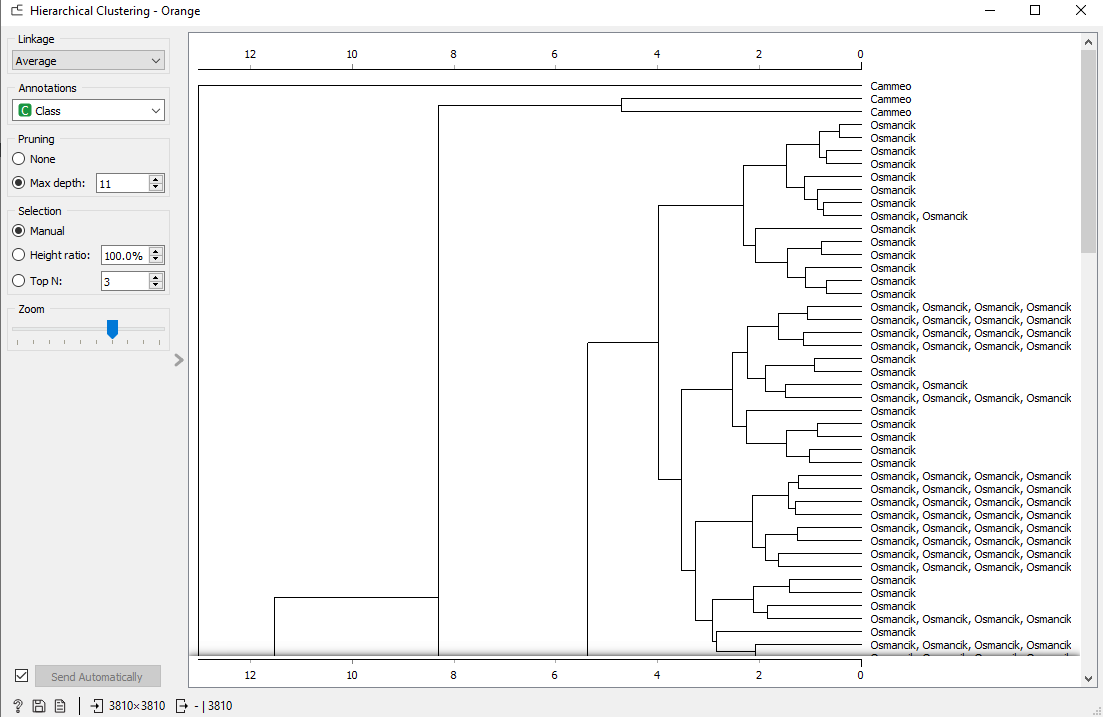
Attēls 2.5

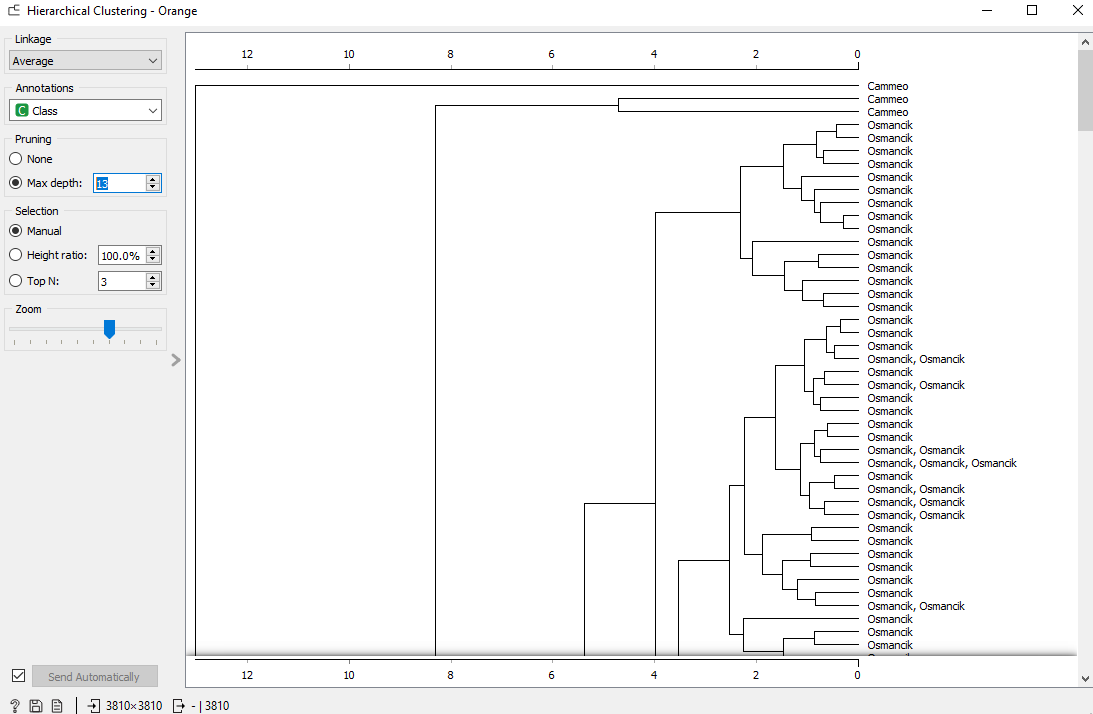
 Attēls 2.5

Saglabājot pēdējo rezultātu konfigurāciju(Attēls 2.5) un tikai nomainot “Pruning” parametru “Max depth” uz vērtību 8(Attēls 2.6), 11(Attēls 2.7) un 13(Attēls 2.8) ir iespējams izmainīts dalīto objektu līmeņu skaitu, ar kuru palīdzību ir viegli apskatīt klašu datu atdalamību.

 Attēls 2.6

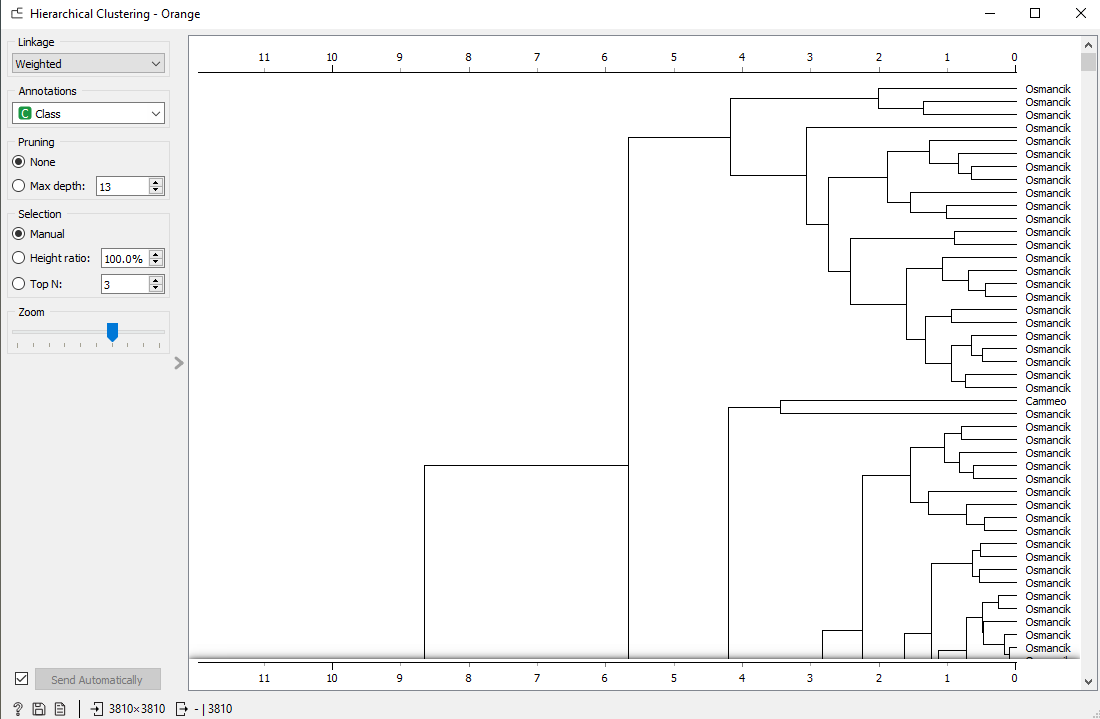
Attēls 2.7

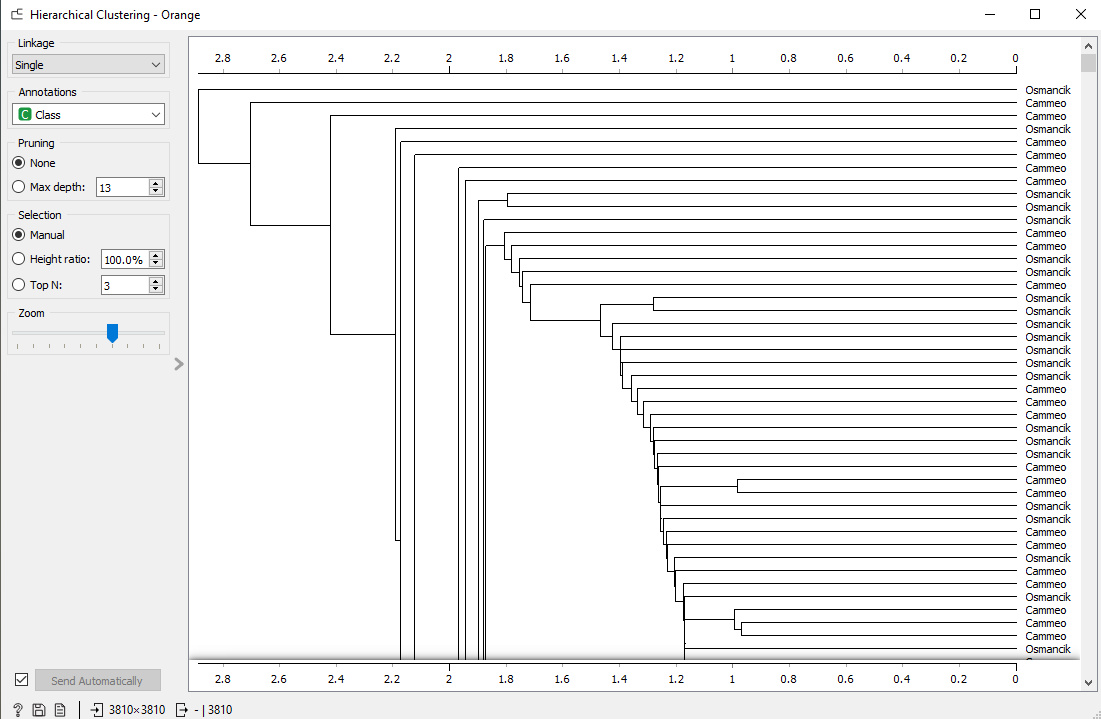




Attēls 2.8

Izmainot “Linkage” funkciju no Average(Attēls 2.5) uz Weigted(Attēls 2.9), dati ir labāk pārskatami pēc sadalījuma nekā ar Average “Linkage”. Pārejot no Weigted uz Signle(Attēls 2.10), dati ir labāk sadalīti nekā Average “Linkage”. Un pārejot uz “Ward”, dati bija ļoti nepārskatāmi(Attēls 2.11).





Attēls 2.10

Attēls 2.9

 Attēls 2.11

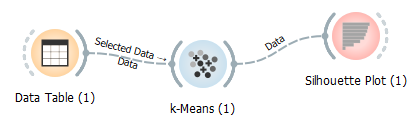
Lai labāk pārskatītu “Ward” iegūtos datus, izmantojam “Height ratio” funkciju, kuru nomainam uz 0.0%, lai labāk apskatītu sadalījumu datiem. (Attēls 2.12)



Attēls 2.12

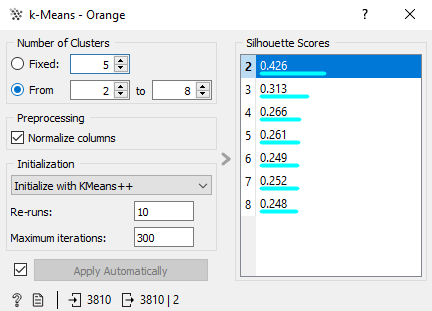
## **2.2 K-vidējo algoritms**

Izmantojot algoritmu datu tabulai jāpievieno “k-Means” logrīks un pie “k-Means” logrīku ir jāpievieno “Silhouette Plot” logrīks, kas palīdzēs atspoguļot rezultātus.(Attēls 2.13)



Attēls 2.13

Atverot “k-Means” logrīku, iegūstam iespēju izmainīt datus. Ir iespējams izmainīt klāsteru skaitu(“Number of Clusters”), šajā gadījumā izmantosim “From” opciju , lai atlasītu specifisku daudzumu klāsteru, manā gadījumā tas būs 2 līdz 8. Ja, izvēlētos “Fixed” opciju, klāsteri būtu fiksētā daudzumā. Izvēloties attiecīgo “Number of Clusters” opciju iegūstam “Silhouette Scores”, iegūtie dati ir labajā pusē logrīkam un tie attēlo datu atdalamību, jo lielāks “Score”, jo labāk atdalāmi dati. Manā gadījumā labākie ir divi klāsteri ar “Score” 0.426.(Attēls 2.14)



“k-Means” hiperparametri un to nozīme(Attēls 2.14):

**Number of Cluster –**

“Fixed”– algoritms klāsterē datus specifiskā daudzumā klāsteru.

“Optimized from “---” to “---” ” – attēlo klāsterēšanas rezultātus izvēlētajam klāsteru diapazonam.

**Processing –**

“Normalize columns” – iespēja normalizēt kolonnas atķeksējot opciju.

**Initialization –**

“Initialize with KMeans++” – pirmais centrs izvēlēts nejauši, pārējie no atpalikušajiem punktiem ar varbūtību proporcionāli kuattālumam no tuvākā centra

“Random initialization” – klāsteriem ir piešķirti nejauši sākotnēji, pēctam atjaunoti ar nākošajām iterācijām.

“Re-runs:” – cik reizies algoritms tiks palaists.

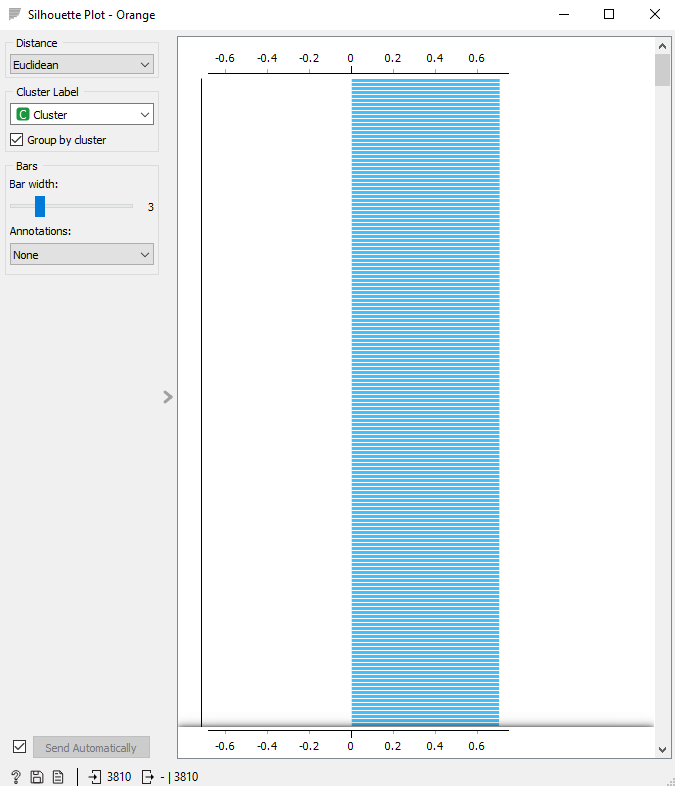
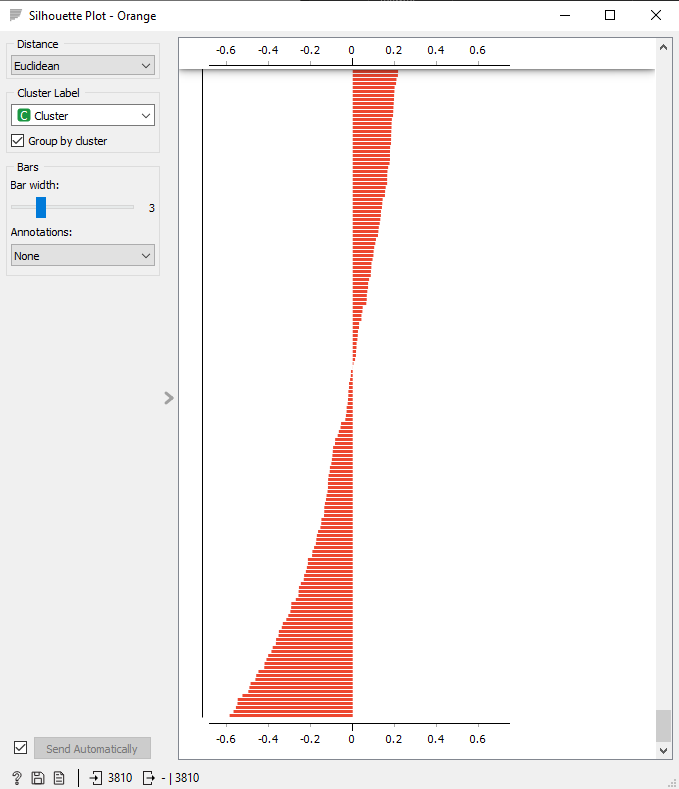
“Maximal iterations:” – maksimālais atkārtojumu skaits katrā algoritmā.

[[**https://orange3.readthedocs.io/en/3.5.0/widgets/unsupervised/kmeansclustering.html#:~:text=k%2DMeans%2B%2B%20(first%20center,distance%20from%20the%20closest%20center)**](https://orange3.readthedocs.io/en/3.5.0/widgets/unsupervised/kmeansclustering.html#:~:text=k%2DMeans%2B%2B%20(first%20center,distance%20from%20the%20closest%20center))](https://orange3.readthedocs.io/en/3.5.0/widgets/unsupervised/kmeansclustering.html#:~:text=k%2DMeans%2B%2B%20(first%20center,distance%20from%20the%20closest%20center)) **- Atsauce**

Attēls 2.14

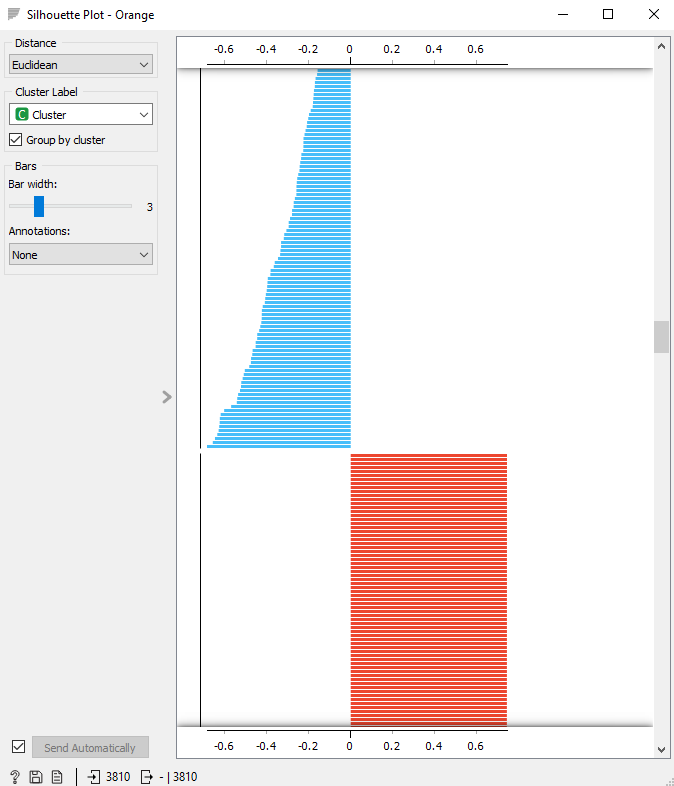
Izmantojot “Silhouette Plot”, ir iespējams grafiski attēlot “k-Means” rezultātus, arī šī opcija mums parādīs datu atdalamību.

Dati, kuri atrodas vairāk uz labo pusi(Attēls 2.15) ir vairāk atdalāmi un dati, kas ir kreisajā pusē ir sliktāk atdalāmi(Attēls 2.16). Objekti, kas atrodas vistālāk kreisajā pusē visticamāk ir nepareizi klasificēti.(Attēls 2.17)

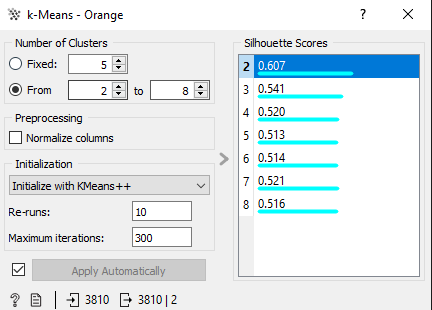
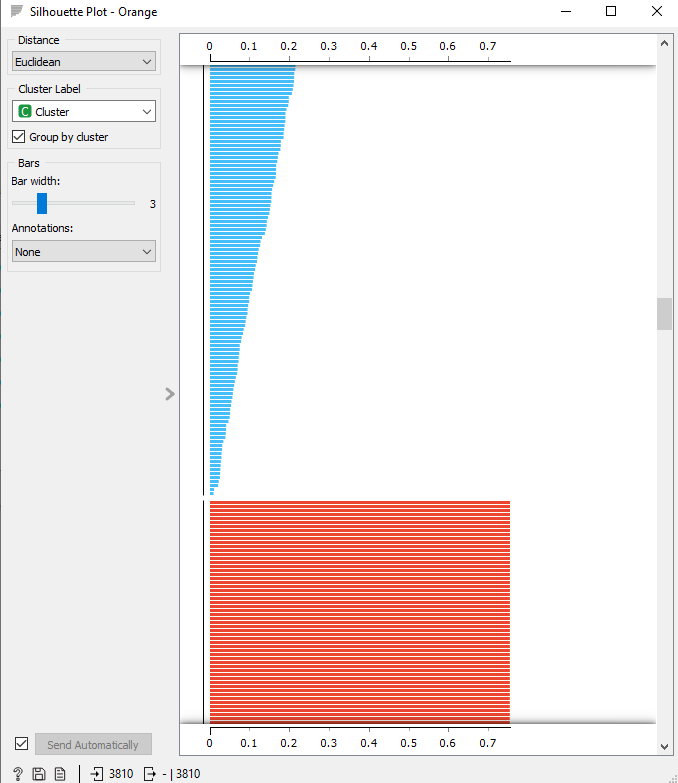


Attēls 2.15 Attēls 2.16

Attēls 2.17



Izmainot “Normalize columns” uz neatķeksētu, vērtējumu paliek krietni tuvāki (Attēls 2.18) un “Silhouette plot” ir krietni grūtāk atšķirt klases atdalījumus(Attēls 2.19).



Attēls 2.18

Attēls 2.19

Balstoties uz abu algoritmu rezultātiem, ir iespējams secināt, ka datu kopā esošās klases ir labi atdalāmas.

# **Pārraudzītā mašīnmācīšanās**

Šim mašīnmācīšanās darbam tika izvēlēti kNN un SVM algoritmi. Darbam tika izvēlēts SVM algoritms, jo tas šķita piemērots datu apstrādei un kNN algoritms tika izvēlēts, jo tas bijis ieprieķš apskatīts nodarbībās un ir pazīstams. Pēc tā pamata principiem kNN ir vienkārši īstenojams un intuitīvi saprotams, kNN algoritmam nav vajadzības no apmācības laika klasifikācijai vai regresijai, jo darbs notiek prognozēšanas laikā.(kNN algoritma priekšrocības, tika pētītas un izmantotas aprakstam no <https://machinelearninginterview.com/topics/machine-learning/how-does-knn-algorithm-work-what-are-the-advantages-and-disadvantages-of-knn/> sadaļas ‘’What are the advantages of KNN ?’’, kur vēl tās tiek uzskaitītas.)

## **3.1 Hiperparametru apraksts**

kNN algoritms(Attēls 3.1):

**Name –** tiek ievadīts nosaukums.

**Neighbors –**

“**Number of neighbors**” – iestatīt tuvāko kaimiņu skaitu, attāluma parametru(metriku) un svarus kā modeļa kritēriju.

“**Metric**” – metrika var būt:

*Eiklīds* – attālums starp 2 punktiem;

*Manheten*a – visu atribūtu absolūto atšķirību summa;

*Maksimālais* – lielākā absolūtā atšķirība starp atribūtiem;

*Mahalanobis* – attālums atarp punktu un sadalījumu.

“**Weight**” – var izvēlēties starp 2 svariem:

*Vienots* – visi punkti katrā apkaimē tiek svērti vienādi.

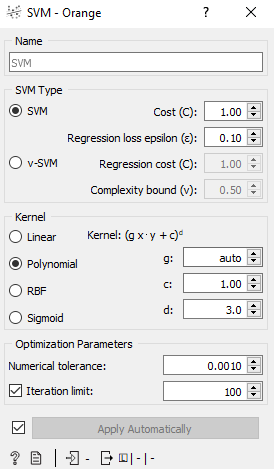
*Attālums* – tuvākiem vaicājuma punkta kaimiņiem ir lielāka ietekme nekā tālākiem kaimiņiem.

**(Izmantotā atsauce uz informāciju** <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/knn.html> **)**

Attēls 3.1

Attēls, kurā ir teksts

Apraksts ģenerēts automātiski

 Attēls 3.2

SVM algoritms (Attēls 3.2):

**SVM Type –**

SVM un v-SVM pamatā ir atšķirīga kļūdas funkcijas samazināšana.

**SVM –**

*Cost(C)* – soda termiņš par zaudējumu. Un tas attiecas uz klasifikācijas un regresijas uzdevumiem.

*Regression loss epsilon (ε)* – defiē attālumu no patiesajām vērtībām, kurā sods nav saistīts ar prognozētajām vērtībām.

**v-SVM –**

*Regression cost (C)* - soda termiņš par zaudējumu. Un tas attiecas tikai uz regresijas uzdevumiem.

*Complexity bound (v*) – apmācības kļūdu daļas augšējā robeža un atbalsta vektoru daļas apakšējā robeža.

**Kernel** –funkcija, kas pārveido atribūtu vietu par jaunu funkciju telpu, lai tā atbilstu maksimālās robežas hiperplaknei. Ir iespējams izvēlēties kāda veida modeli veidot.

*g* – gamma konstante kodola funkcijā,

*c* - konstante c0 kodola funkcijā,

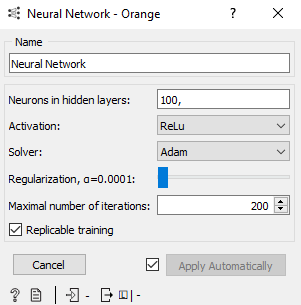
*d* – kodola pakāpe, kuras noklusējums ir 3.

**Optimization Parameters –**

*Numerical tolerance* – atļautā novirze no paredzamās vērtības

*Iteration limit* – iterāciju ierobežojums.

**(Izmantotā atsauce uz informāciju** <https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/svm.html> **)**

 Attēls 3.3

Mākslīgais neironu tīkls (Attēls 3.3):

**Neurons in hidden layers –** i-tais elementrs,apzīmē neironu skaitu i-tajā slēptajā slānī.

**Activation –** slēptā slāņa aktivizēšanas funkcija,

*Identity* – bezoperācijas aktivizēšana, noderīga lineāro sašaurinājumu īstenošanai.

*Logistic* – loģistikas sigmoīda funkcija.

*Tanh* – hiperboliskā tangensa funkcija

*ReLu* – rektificētas lineāras vienības funkcija.

**Solver –**

*L-BFGS-B* **–** optimizētājs kvaziņūtona metožu saimē.

*SGD* – stohastiskā gradienta nolaišanās.

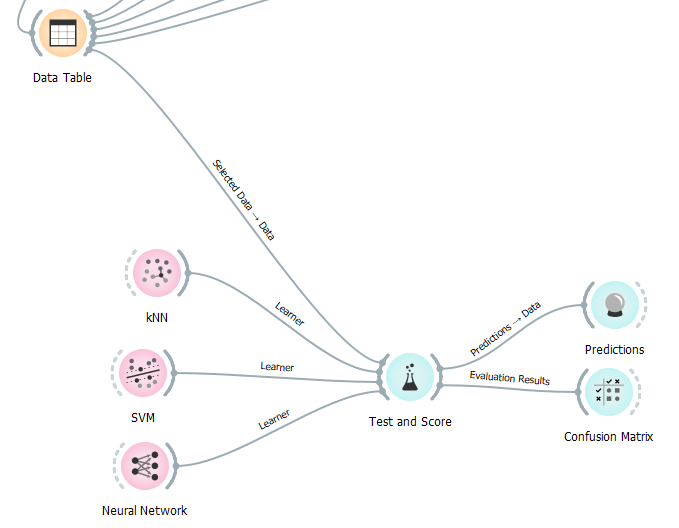
*Adam* – uz stohastisko gradientu balstīts optimizētājs

**Regularization (Alpha) –** soda parametrs.

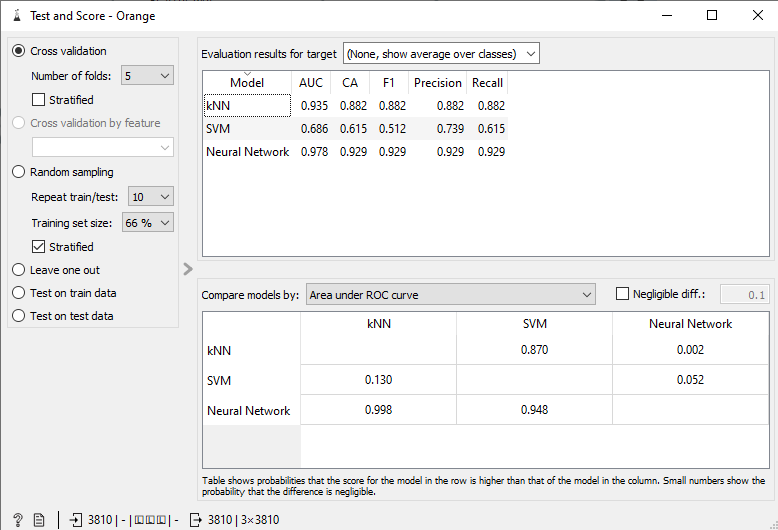
**Maximal number of iterations –** maksimālais iterāciju skaits.

**(Izmantotā atsauce uz informāciju** [**https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/neuralnetwork.html**](https://orange3.readthedocs.io/projects/orange-visual-programming/en/latest/widgets/model/neuralnetwork.html) **)**

Attēls 3.4

 Darbam ar algoritmiem ir nepieciešams sākumā savienot logrīkus Data Table, kNN, SVM, Neural Network, Test and Score, Confusion Matrix un Predictions (Attēls 3.4)

Sākotnējie dati ar noklusējuma vērtībām un hiperparametriem(Attēls 3.5, Attēls 3.6 un Attēls 3.7)

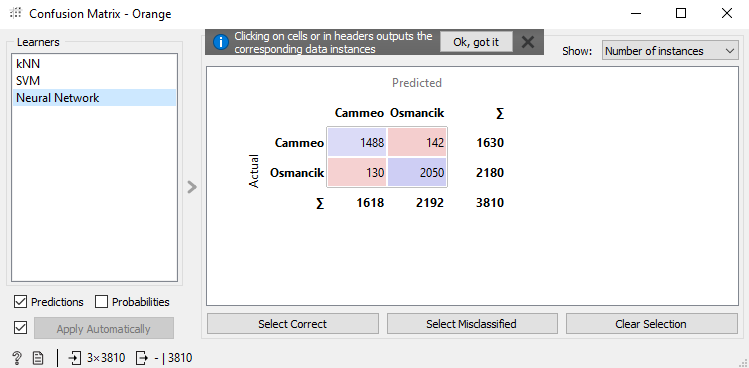


Attēls 3.5

Attēls, kurā ir galds

Apraksts ģenerēts automātiski

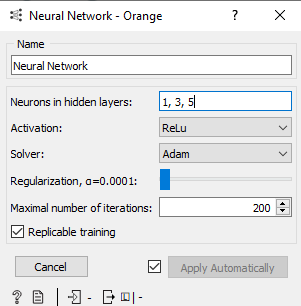
Attēls 3.6



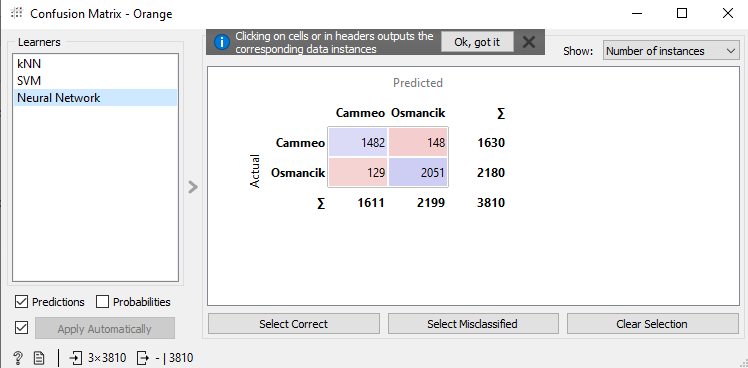
Attēls 3.7

Sākotnēji tiek izmainīts neironu skaits slēptajos slāņos uz “1, 3, 5”(Attēls 3.8) un apskatīts kā izmainijušies rezultāti(Attēls 3.9 un Attēls 3.10).

Attēls 3.8 Attēls 3.10

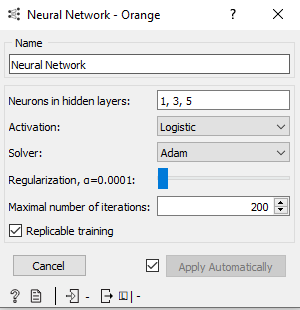
Attēls, kurā ir galds

Apraksts ģenerēts automātiski

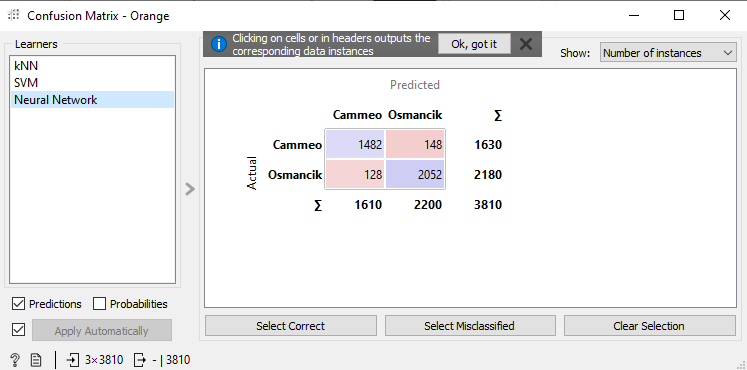


Attēls 3.9

Tālāk tam pašam algoritmam tiek izmainīta “ReLu” akticācija uz “Logistic” (Attēls 3.11), atkal attēlojot iegūtos rezultātus (Attēls 3.12 un Attēls 3.13).

Attēls 3.11

Attēls, kurā ir galds

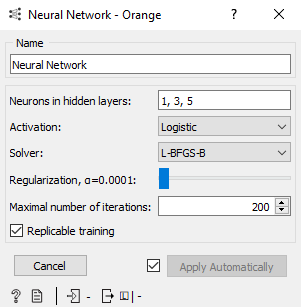
Apraksts ģenerēts automātiski 

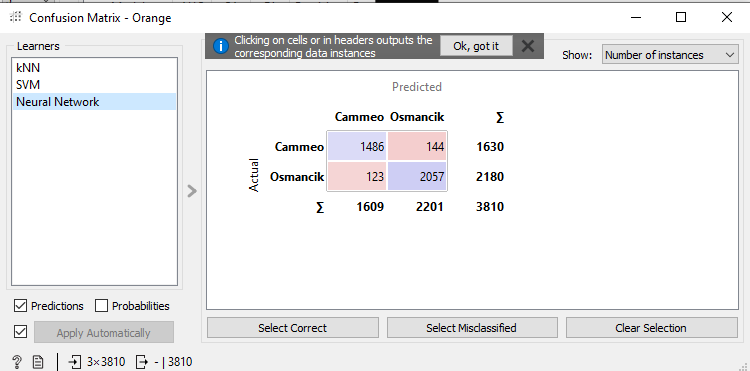
Attēls 3.12

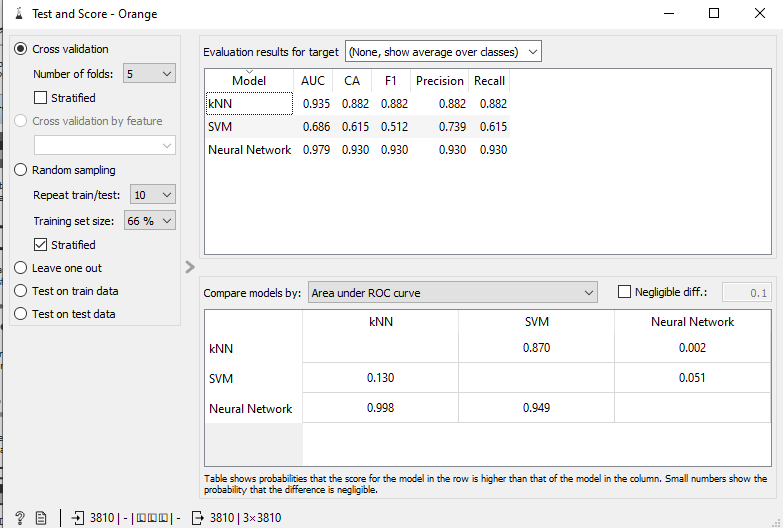
Pārejot no ReLu uz Logistic, redzams, ka CA un F1 vērtības izmainās par vērtību 0.001, minimāli uzlabojot rezultātu(Attēls 3.13). Un Confusion matrix rezultāti uzlabojās par 1 vērtību nepareizo vērtību.(Attēls 3.12).

Attēls 3.13

Tālāk nomainam “Solver” no “Adam” uz “L-BFGS-B”(Attēls 3.14), atspoguļojot rezultātus(Attēls 3.15 un Attēls 3.16) , ir iespējams novērot, ka palielinājās rezultātu daudzums vērtējumu par mazu daudzumu Confusion Matrix un Test and Score logrīkos.

Attēls 3.14

 Attēls 3.15

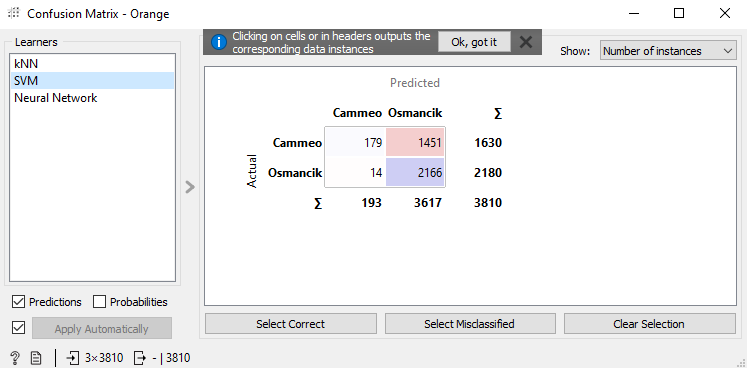
 Attēls 3.16

Izmantotās hiperparametru vērtības Neural Netowork algoritmam(Tabula 3.1).

|  |  |  |  |
| --- | --- | --- | --- |
| n.p.k | Neurons in hidden layers | Activation | Solver |
| 1 | 1,3,5 | ReLu | Adam |
| 2 | 1,3,5 | Logistic | Adam |
| 3 | 1,3,5 | Logistic | L-BFGS-B |

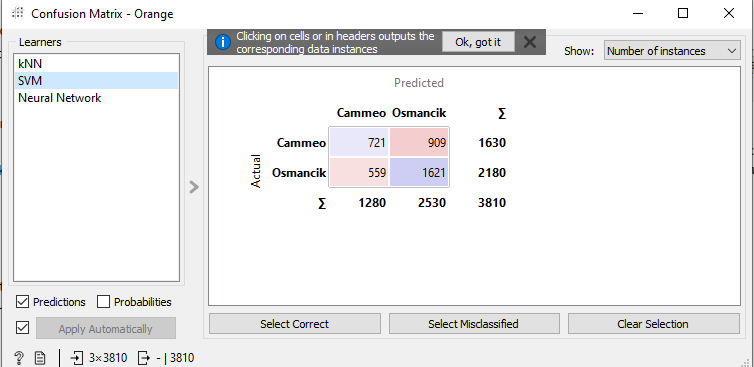
Tabula 3.1

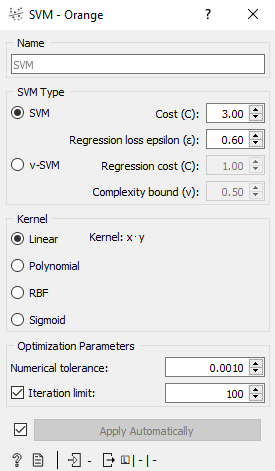
Tālāk tiks mainīti hiperparametri SVM algoritmam, lai to rezultātus, salīdzinātu ar sākotnējiem rezultātiem(Attēls 3.17)

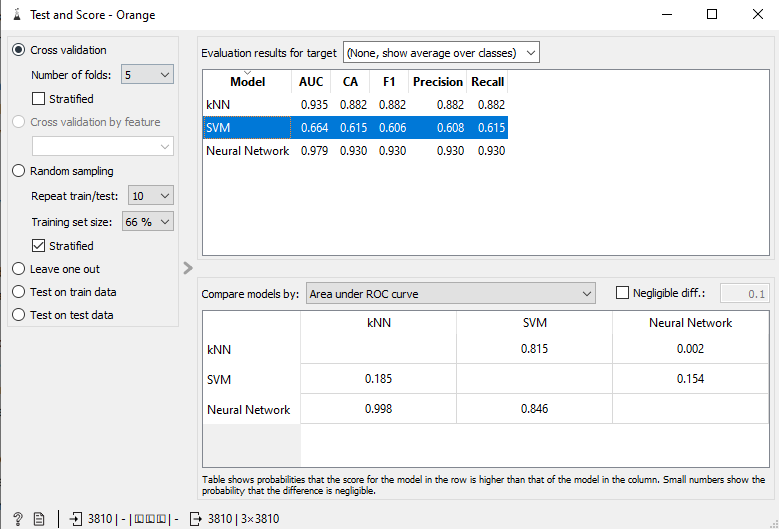


Attēls 3.17

Tiek izmainīti SVM hiperparametri Cost uz 3, Regression loss epsilon uz 0.6 un Kernel uz Linear(Attēls 3.18), attēloti to rezultāti(Attēls 3.19 un Attēls 3.20)



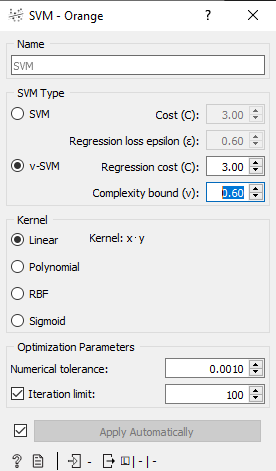
 Attēls 3.18

Attēls 3.19

Attēls 3.20

Iespējams novērot, ka samazinājās nepareizi klasificēto elementu skaits(Attēls 3.20)

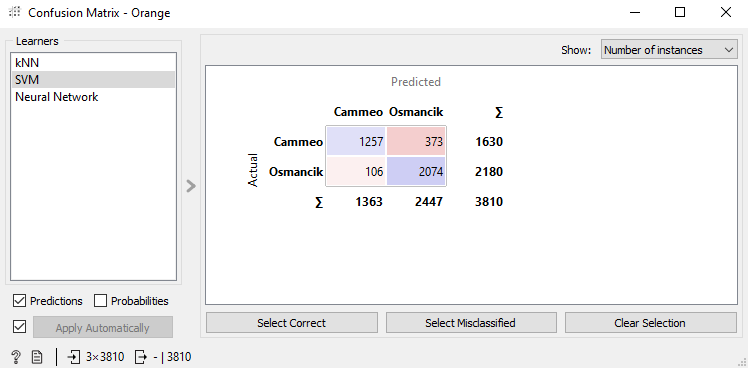
Tālāk algoritmam SVM Type tiek nomainīts no SVM uz v-SVM iestatot Regression cost uz 3,00 un Complexity bound vērtību uz 0,6 un atstāj Kernel uz “Linear”(Attēls 3.21), datus attēlo(Attēls 3.22 un Attēls 3.23)



Attēls 3.21 Attēls 3.22

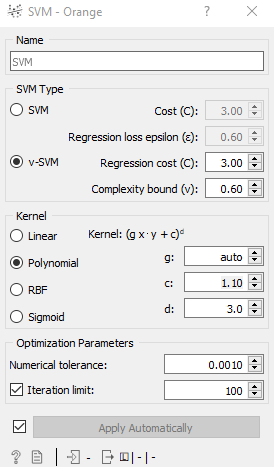
Attēls, kurā ir teksts

Apraksts ģenerēts automātiski

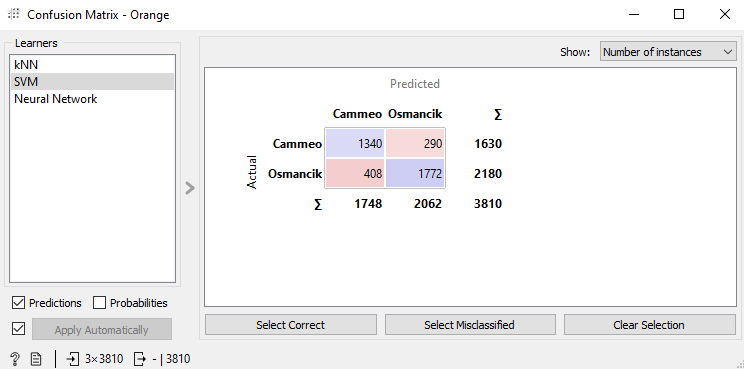
 Attēls 3.23

Redzams, ka paaugstinājies nepareizi un pareizi klasificēto elementu skaits(Attēls 3.23). Precizitātes novērtējums krietni paaugstinājās(Attēls 3.22).

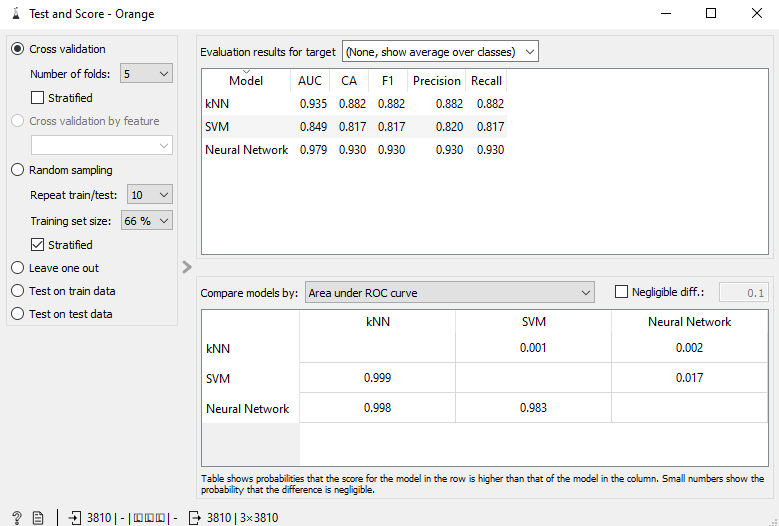
Beidzot, SVM algoritmam tiek nomainīts Kernel uz “Polynomial”, g vērtība “auto”, c vērtība uz 1.10 un d vērtība uz 3.0(Attēls 3.24), atspoguļo rezultātus (Attēls 3.25 un Attēls 3.26)

Attēls 3.24

Attēls 3.25



Attēls 3.26



Izmainot vērtības redzams, ka precizitātes novērtējums samazinājās(Attēls 3.26).

Samazinājās nepareizi klasificētie elementi(Attēls 3.25).

Izmantotās hiperparametru vērtības SVM algoritmam(Tabula 3.2).

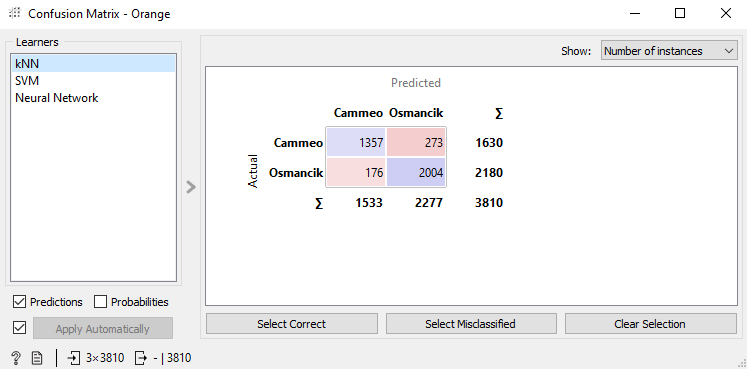
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n.p.k | Cost | Regresion loss epsilon | Kernel | SVM Type | Regression cost | Complexity bound |
| 1 | 3 | 0.60 | Linear | SVM | - | - |
| 2 | - | - | Linear | v-SVM | 3.00 | 0.60 |
| 3 | - | - | Polynomial | v-SVM | 3.00 | 0.60 |

Tabula 3.2

Pēdējā algoritma kNN sākotnējās(Attēls 3.5, Attēls 3.27 un Attēls 3.28) vērtības tiek izmainītas un salīdzināti rezultāti.

Attēls 3.27 Attēls 3.28

Attēls, kurā ir teksts

Apraksts ģenerēts automātiski

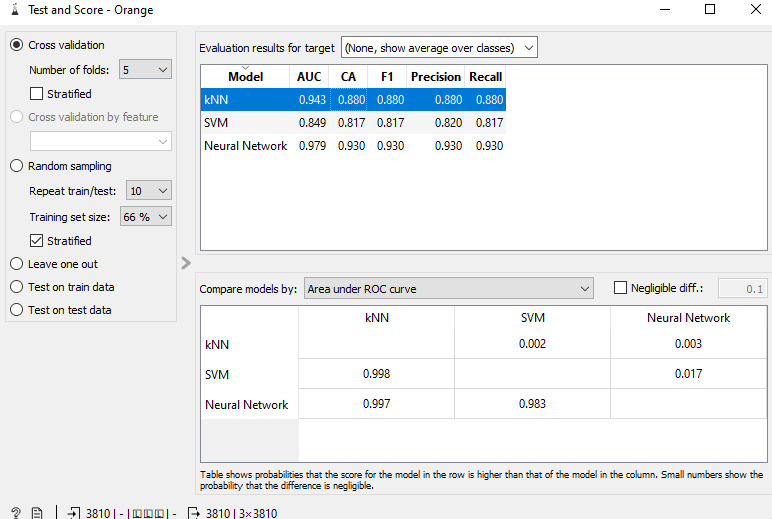
Tiek izmainīta vērtība Number of neighbours uz 12(Attēls 3.29), apskatamies rezultātus.(Attēls 3.30 un 3.31)

Attēls 3.29

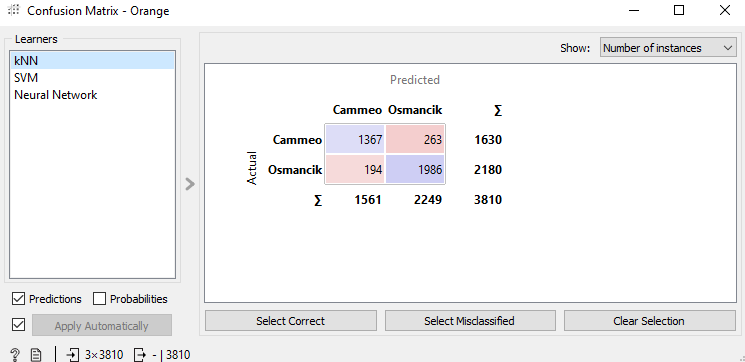
Attēls, kurā ir teksts

Apraksts ģenerēts automātiski

Attēls 3.30

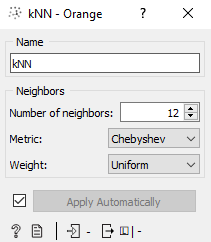


Attēls 3.31



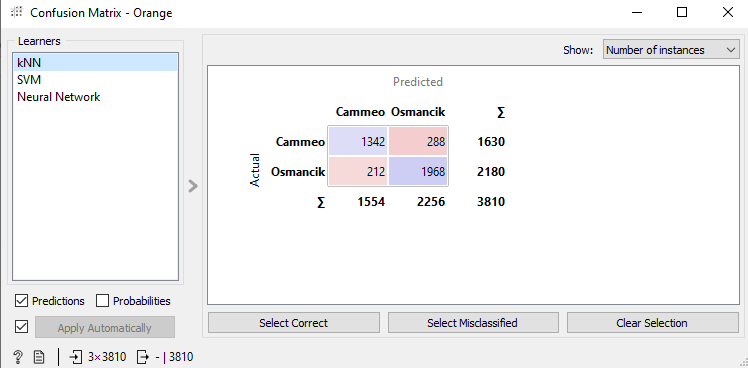
Mazliet samazinājās nepareizi klasificētie elementi(Attēls 3.31). Precizitātes vērtējums mazliet pasliktinājās(Attēls 3.30).

Tālāk Metric vērtību nomaina uz Chebyshev(Attēls 3.32) un attēlo rezultātus(Attēls 3.33 un Attēls 3.34).

 Attēls 3.32

Attēls, kurā ir galds

Apraksts ģenerēts automātiskiAttēls 3.33

Attēls 3.34

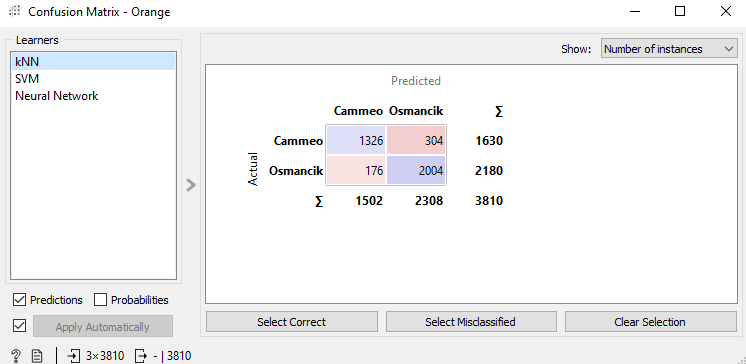
Precizitātes vērtējums mazliet pasliktinājās(Attēls 3.33) un samazinājās nepareizi klasificētie vērtējumi (Attēls 3.34)/

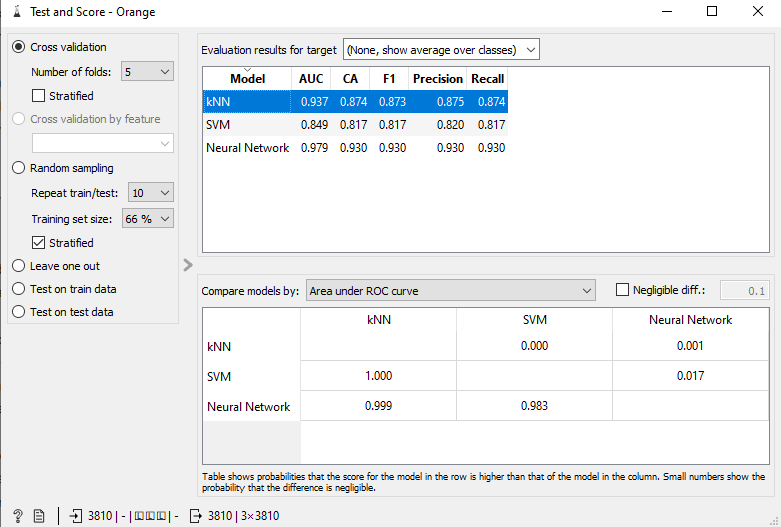
Pēdējais tiek nomainīts hiperparametrs Weight uz “Distance”(Attēls 3.35) un aplūkoti dati(Attēls 3.36 un Attēls 3.37).

Attēls 3.35

Attēls, kurā ir teksts

Apraksts ģenerēts automātiski

 Attēls 3.36

 Attēls 3.37 Paaugstinājās nepareizi klasificēto datu skaits un samazinājās pareizi klasificēto skaits(Attēls 3.36). Precizitātes vērtējums mazliet paaugstinājās.

Izmantotās hiperparametru vērtības kNN algoritmam(Tabula 3.3)

|  |  |  |  |
| --- | --- | --- | --- |
| n.p.k | Number of neighbors | Metric | Weight |
| 1 | 12 | Euclidean | Uniform |
| 2 | 12 | Chebyshev | Uniform |
| 3 | 12 | Chebyshev | Distance |

Tabula 3.3

## **3.2 Secinājumi**

Neural network algoritmam nomainot paslēpto neironu vērtības, mazliet palielināja pareizi un nepareizi klasificētās vērtības. Vislielākās izmaiņas tika iegūtas izmainot solver vērtību no “Adam” uz “L-BFGS-B”, šīs izmaiņas uzlaboja precizitātes novērtējumu algoritmam un mazliet izmainīja klasificētos datus matricā.

SVM algoritmam izmainot Cost uz 3, Regression loss epsilon uz 0.6 un Kernel uz Linear, tika iegūtas lielākās izmaiņas rezultātos, precizitātes vērtējums nokritās un krietni palielinājās pareizi klasificētās vērtības, samazinot nepareizi klasificētās. Vislabākie precizitātes vērtējumi tika iegūti nomainot SVM Type no SVM uz v-SVM iestatot Regression cost uz 3,00 un Complexity bound vērtību uz 0,6, precizitātes vērtējums krietni uzlabojās. Nomainot Kernel uz “Polynomial”, samazinājās precizitātes vērtējums, bet uzlabojās pareizi klasificēto vērtību skaits.

kNN algoritmam izmainot hiperparametru netika iegūts labāks precizitātes vērtējums, salīdzinot ar sākotnējo vērtējumu, bet nomainot Metric vērtību uz Chebyshev tika iegūts visaugstākais pareizi klasificēto vērtību skaits un mazākais nepareizi klasificēto vērtību skaits.

Balstoties uz rezultātiem, ir iespējams secināt, ka šajā gadījumā labākais algoritms manai datu kopai būtu SVM algoritms.