RIGA TECHNICAL UNIVERSITY

Faculty of Computer Science, Information Technology and Energy

Report on the second practical assignment

Study course "Fundamentals of artificial intelligence"

Team number: 5
Students:
Pooja Odedara , 221ADB033

Mykyta Medvediev 221ADB084

Yamben Paul-Joseph 240AEB035

Muhammad Ali Tursunmurodov 213AEB032

Hadjadj Adnane 240ADB003

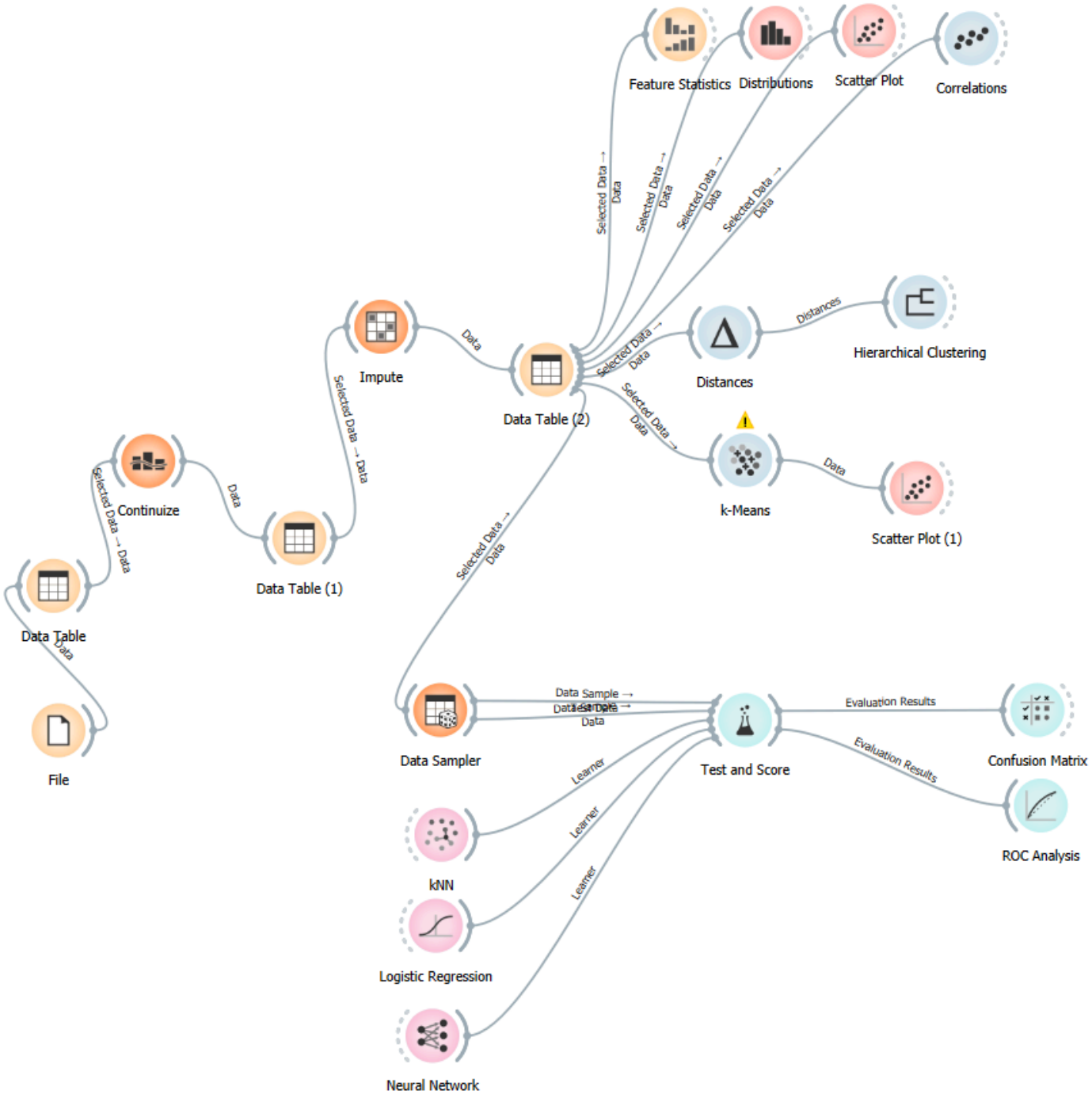
Mariana Mechyk 221ADB094

Teaching staff: Alla Anohina-Naumeca

Project link: Team 5 Practical Task 2
Link to dataset: Dry-Bean dataset

Orange tool workflow

<screenshot describing Part 1, 2 and 3 workflow>



Part I

Description of the dataset

Dataset title: ***Dry Bean***

Dataset source: UCI Machine Learning Repositories DOI

Creator and/or owner of the dataset:

The “Dry Bean” dataset was created by Murat KOKLU from the Faculty of Technology at Selcuk University in Turkey

Description of the dataset problem domain:

Seven different types of dry beans were used in this research, taking into account the features such as form, shape, type, and structure by the market situation. A computer vision system was developed to distinguish seven different registered varieties of dry beans with similar features in order to obtain uniform seed classification. For the classification model, images of 13,611 grains of 7 different registered dry beans were taken with a high-resolution camera. Bean images obtained by the computer vision system were subjected to segmentation and feature extraction stages, and a total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains.

Information about the method or procedure for collecting the dataset:

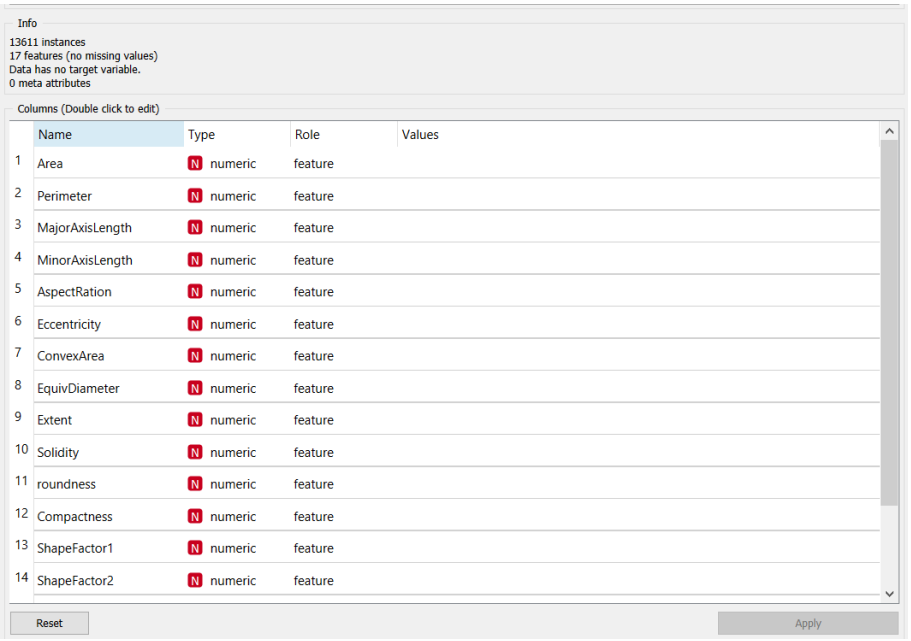
The “Dry Bean” dataset was collected using a high-resolution camera to take photographs of 13,611 beans of seven different varieties

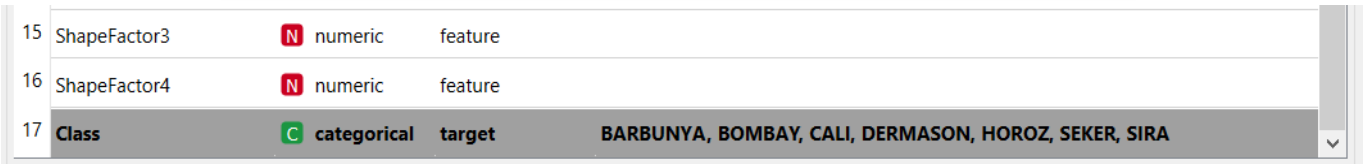
Description of the dataset content

Number of data objects in the dataset: **13611**

Number of classes in the dataset: **7**

Representation of features (attributes) of the dataset together with their roles in the Orange tool:





Description of classes:

<labels used to represent classes and the meaning of each class; if the dataset provides several possible data classifications, then the report should clearly identify which classification is being addressed in the assignment>

Number of data objects belonging to each class:

Number of rows is the number of instances. Using COUNTIF() gets the number of rows where class label is met.

Class label	Number of data objects
SEKER	2027
BARBUNYA	1332
BOMBAY	522
CALI	1630
HOROZ	1928
SIRA	2636
DERMASON	3546

Description of features:

<add rows to table as needed>

Feature title	Explanation of the feature	Value type	Range of values

Area	The area of a bean zone and the number of pixels within its boundaries	Integer	20000-254616
Perimeter	Bean circumference is defined as the length of its border.	Continuous	524.76- 1985.37
MajorAxisLength	The distance between the ends of the longest line that can be drawn from a bean	Continuous	183.601- 738.86
MinorAxisLength	The longest line that can be drawn from the bean while standing perpendicular to the main axis	Continuous	122.51 - 460.19
AspectRation	Defines the relationship between MajorAxisLength and MinorAxisLength	Continuous	1.024 - 2.43
Eccentricity	Eccentricity of the ellipse having the same moments as the region	Continuous	0.2 - 1
ConvexArea	Number of pixels in the smallest convex polygon that can contain the area of a bean seed	Integer	20684 - 263261
EquivDiameter	Equivalent diameter: The diameter of a circle having the same area as a bean seed area	Continuous	161.24 - 570
Extent	The ratio of the pixels in the bounding box to the bean area	Continuous	0.55 - 0.9
Solidity	Also known as convexity. The ratio of the pixels in the convex shell to those found in beans.	Continuous	0.9 - 1
roundness	Calculated with the following formula: (4piA)/(P^2)	Continuous	0.48 -1
Compactness	Measures the roundness of an object	Continuous	0.64 - 1
ShapeFactor1	-	Continuous	0.00277 - 0.02

ShapeFactor2	-	Continuous	0.000564 - 0.004
ShapeFactor3	-	Continuous	0.41 - 1
ShapeFactor4	-	Continuous	0.948 - 1

Data file structure:

Screenshot from Orange Tool Data Table Widget describing some instances of SEKER class:

	Class	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRation	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness
1	SEKER	28395	610.291	208.178	173.889	1.19719	0.549812	28715	190.141	0.763923	0.988856	0.958027
2	SEKER	28734	638.018	200.525	182.734	1.09736	0.411785	29172	191.273	0.783968	0.984986	0.887034
3	SEKER	29380	624.11	212.826	175.931	1.20971	0.562727	29690	193.411	0.778113	0.989559	0.947849
4	SEKER	30008	645.884	210.558	182.517	1.15364	0.498616	30724	195.467	0.782681	0.976696	0.903936
5	SEKER	30140	620.134	201.848	190.279	1.0608	0.33368	30417	195.897	0.773098	0.990893	0.984877
6	SEKER	30279	634.927	212.561	181.51	1.17107	0.520401	30600	196.348	0.775688	0.98951	0.943852
7	SEKER	30477	670.033	211.05	184.039	1.14677	0.489478	30970	196.989	0.762402	0.984081	0.85308
8	SEKER	30519	629.727	212.997	182.737	1.16559	0.51376	30847	197.124	0.770682	0.989367	0.967109
9	SEKER	30685	635.681	213.534	183.157	1.16585	0.514081	31044	197.66	0.771561	0.988436	0.95424
10	SEKER	30834	631.934	217.228	180.897	1.20083	0.553642	31120	198.139	0.783683	0.99081	0.970278

Screenshot from Orange Tool Data Table Widget describing some instances of BARBUNYA and BOMBAY classes:

3346	BARBUNYA	102015	1271.97	456.792	286.894	1.5922	0.778162	103901	360.402	0.804973	0.981848	0.792356
3347	BARBUNYA	102379	1296.38	456.722	286.558	1.59382	0.778679	104111	361.044	0.739412	0.983364	0.765523
3348	BARBUNYA	105542	1265.62	466.136	288.999	1.61293	0.78461	107112	366.579	0.747888	0.985342	0.827993
3349	BARBUNYA	115967	1359.76	449.455	331.305	1.35662	0.675755	118497	384.257	0.742712	0.978649	0.788166
3350	BOMBAY	114004	1279.36	451.361	323.748	1.39417	0.696795	115298	380.991	0.748987	0.988777	0.87528
3351	BOMBAY	117034	1265.93	425.924	351.215	1.21271	0.565722	118019	386.021	0.746319	0.991654	0.91771
3352	BOMBAY	126503	1326.96	475.772	339.382	1.40188	0.70083	128220	401.334	0.771313	0.986609	0.902809
3353	BOMBAY	128118	1360.13	504.025	325.678	1.54762	0.763206	129274	403.887	0.782247	0.991058	0.870274

Screenshot from Orange Tool Data Table Widget describing some instances of CALI and HOROZ classes:

5495	CALI	106064	1234.6	481.098	282.353	1.70389	0.809665	106961	367.485	0.810032	0.991614	0.874425
5496	CALI	106700	1255.06	469.423	291.873	1.60831	0.7832	108228	368.585	0.696466	0.985882	0.85123
5497	CALI	106806	1263.9	494.727	276.176	1.79134	0.829679	108109	368.768	0.705451	0.987947	0.840196
5498	CALI	107911	1298.82	498.598	279.35	1.78485	0.828309	110337	370.67	0.731898	0.978013	0.803852
5499	CALI	114858	1300.82	512.737	287.562	1.78305	0.827926	115826	382.416	0.688713	0.991643	0.852977
5500	CALI	115608	1298.62	500.298	296.899	1.68508	0.804876	117222	383.662	0.77965	0.986231	0.861453
5501	CALI	116272	1326.58	534.484	279.783	1.91035	0.852048	118144	384.762	0.766551	0.984155	0.830264
5502	HOROZ	33006	710.496	283.02	149.624	1.89155	0.848829	33354	204.999	0.635476	0.989566	0.821636
5503	HOROZ	33263	719.325	271.339	158.566	1.7112	0.811477	34108	205.795	0.672741	0.975226	0.807832
5504	HOROZ	33407	706.222	281.535	152.742	1.84321	0.840035	33989	206.24	0.809867	0.982877	0.841715
5505	HOROZ	33518	702.956	277.571	154.306	1.79884	0.83124	34023	206.583	0.808383	0.985157	0.852377
5506	HOROZ	33518	702.956	277.571	154.306	1.79884	0.83124	34023	206.583	0.808383	0.985157	0.852377

Screenshot from Orange Tool Data Table Widget describing some instances of SIRA and DERMASON classes:

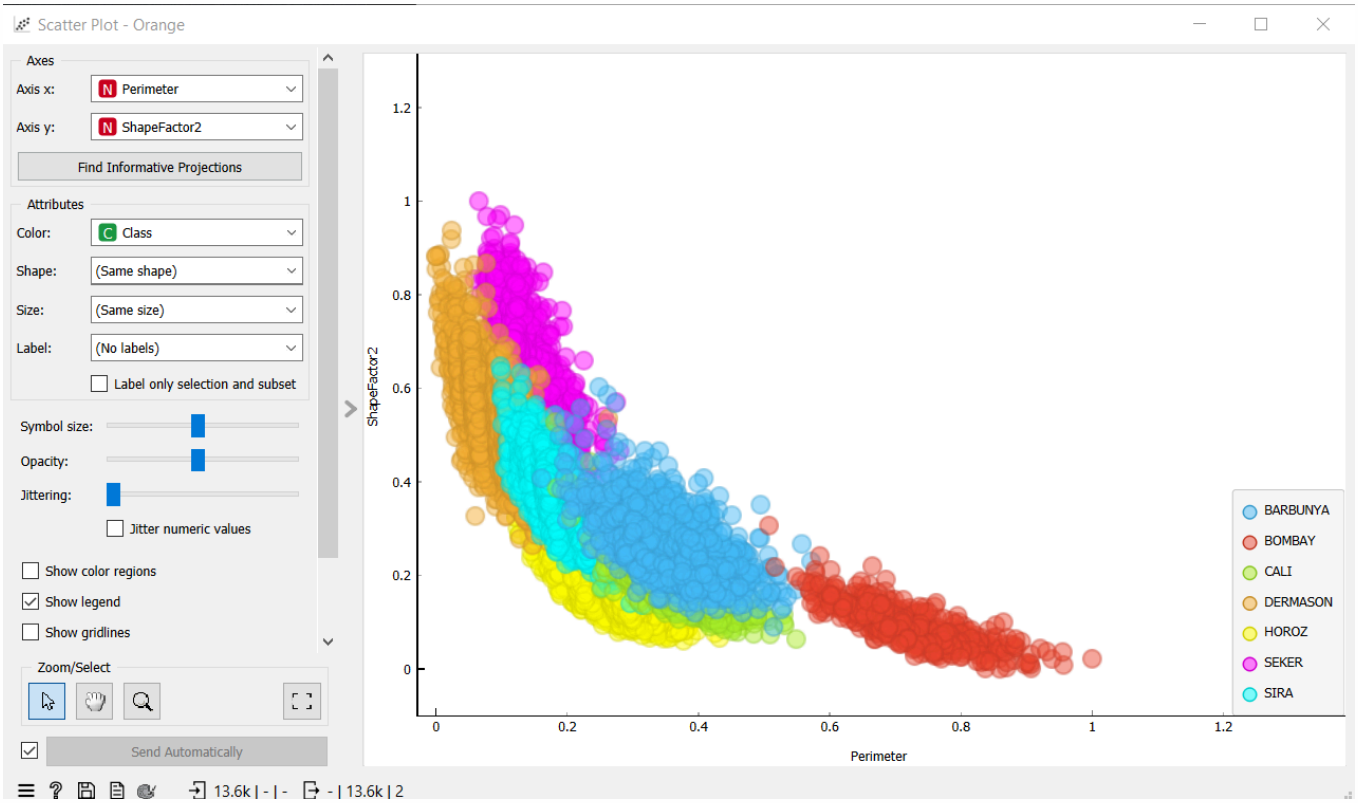
10061	SIRA	58063	941.882	366.69	202.791	1.80821	0.83316	58936	271.897	0.735012	0.985187	0.822463
10062	SIRA	58074	910.115	351.959	210.418	1.67267	0.80161	58609	271.923	0.777648	0.990872	0.881047
10063	SIRA	59431	956.785	390.489	194.565	2.00699	0.867028	60276	275.082	0.689839	0.985981	0.81582
10064	SIRA	60493	931.321	363.814	212.614	1.71115	0.811464	61239	277.529	0.791814	0.987818	0.876428
10065	SIRA	63612	984.282	400.931	204.348	1.96201	0.860362	64581	284.593	0.815664	0.984996	0.825106
10066	DERMASON	20420	524.932	183.601	141.886	1.294	0.634655	20684	161.244	0.790187	0.987237	0.931235
10067	DERMASON	20464	528.408	191.249	136.368	1.40245	0.701123	20772	161.417	0.747407	0.985172	0.921004
10068	DERMASON	20548	524.736	183.965	142.672	1.28942	0.631299	20825	161.748	0.759686	0.986699	0.937773
10069	DERMASON	20711	525.413	186.079	142.082	1.30966	0.645743	20988	162.389	0.793525	0.986802	0.942778

Information about missing values or outliers:

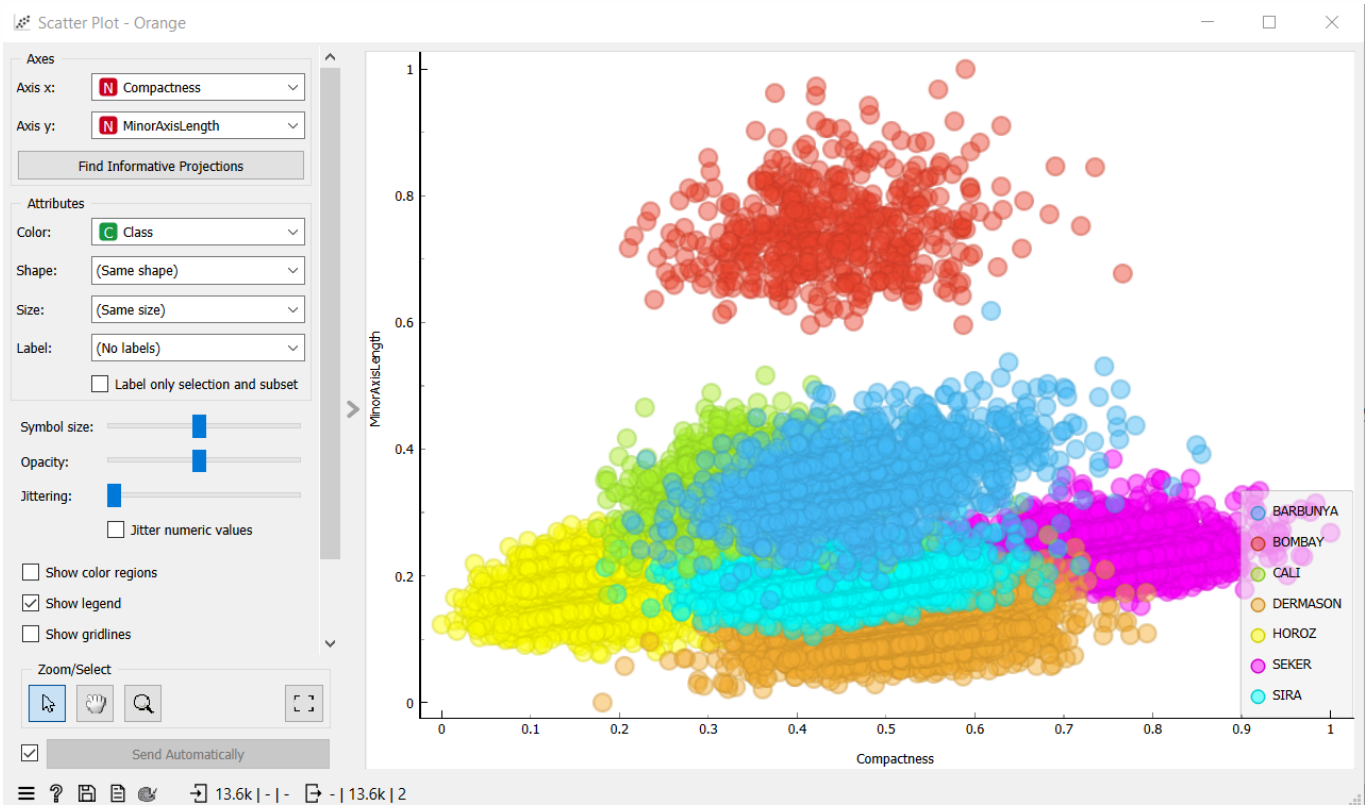
Given dataset has no missing values.

Visual and statistical representation of the dataset

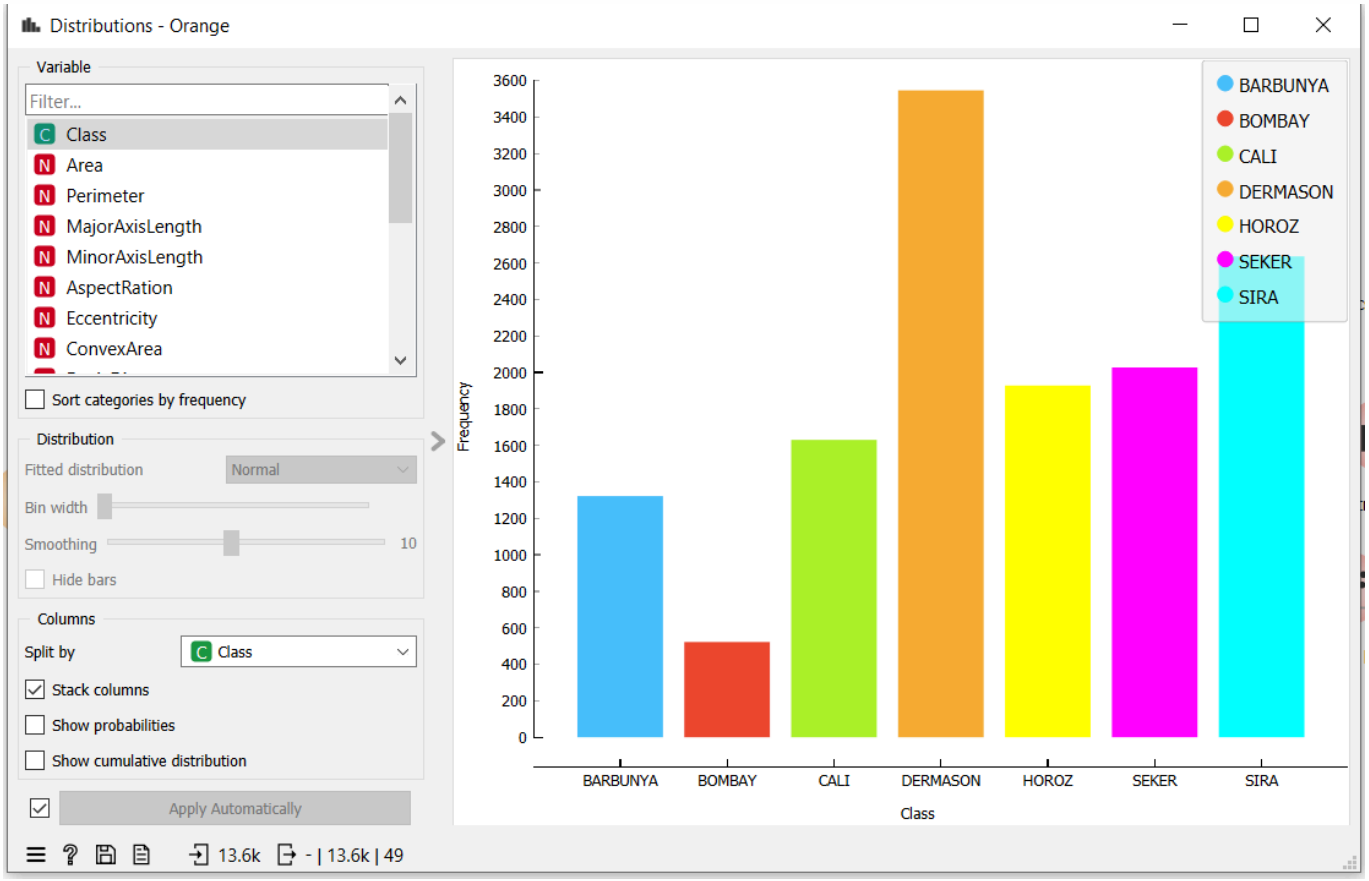
Scatter plot illustrating class separability based on Perimeter and ShapeFactor2



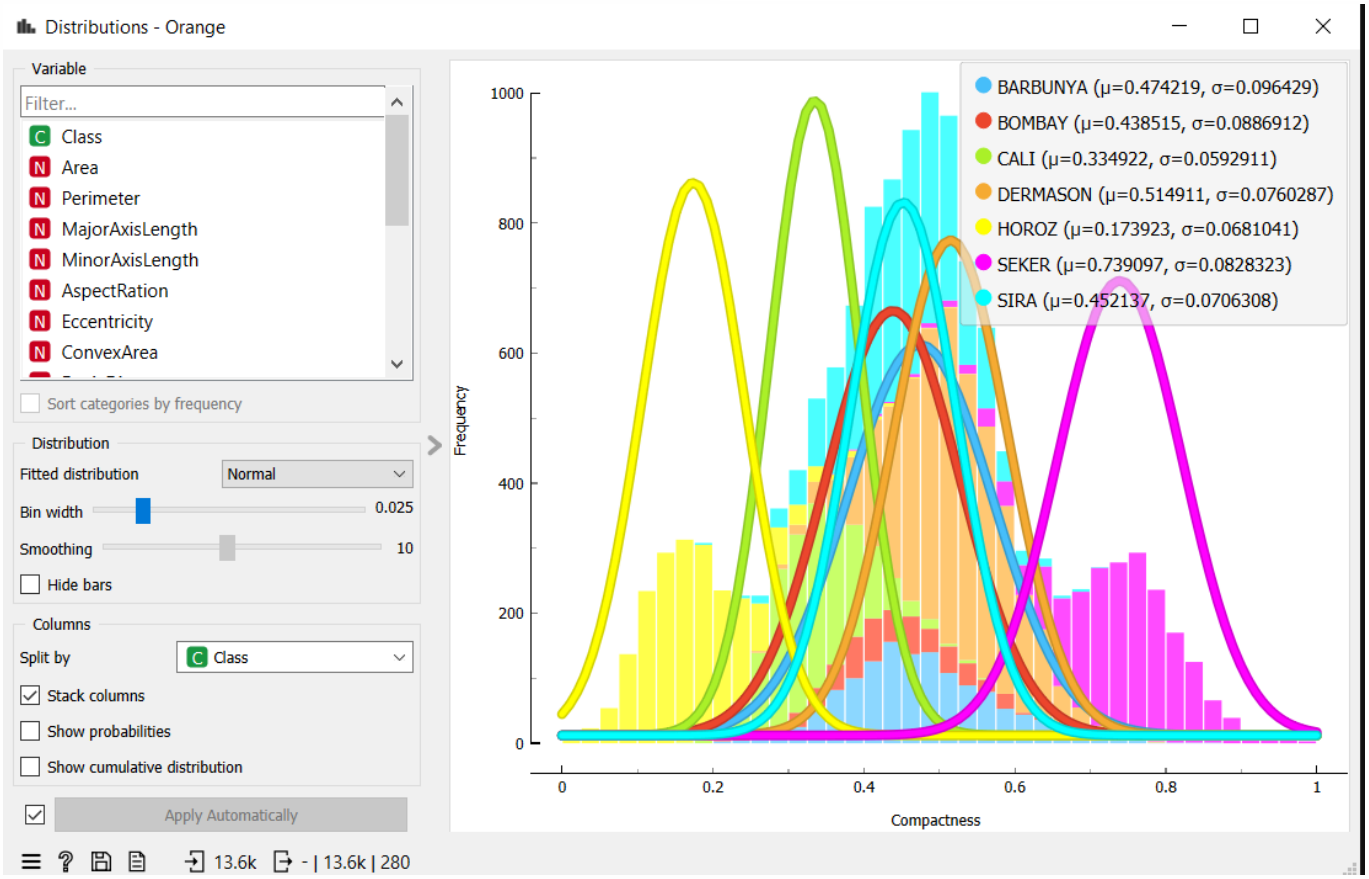
Scatter plot illustrating class separability based on Compactness and MinorAxisLength



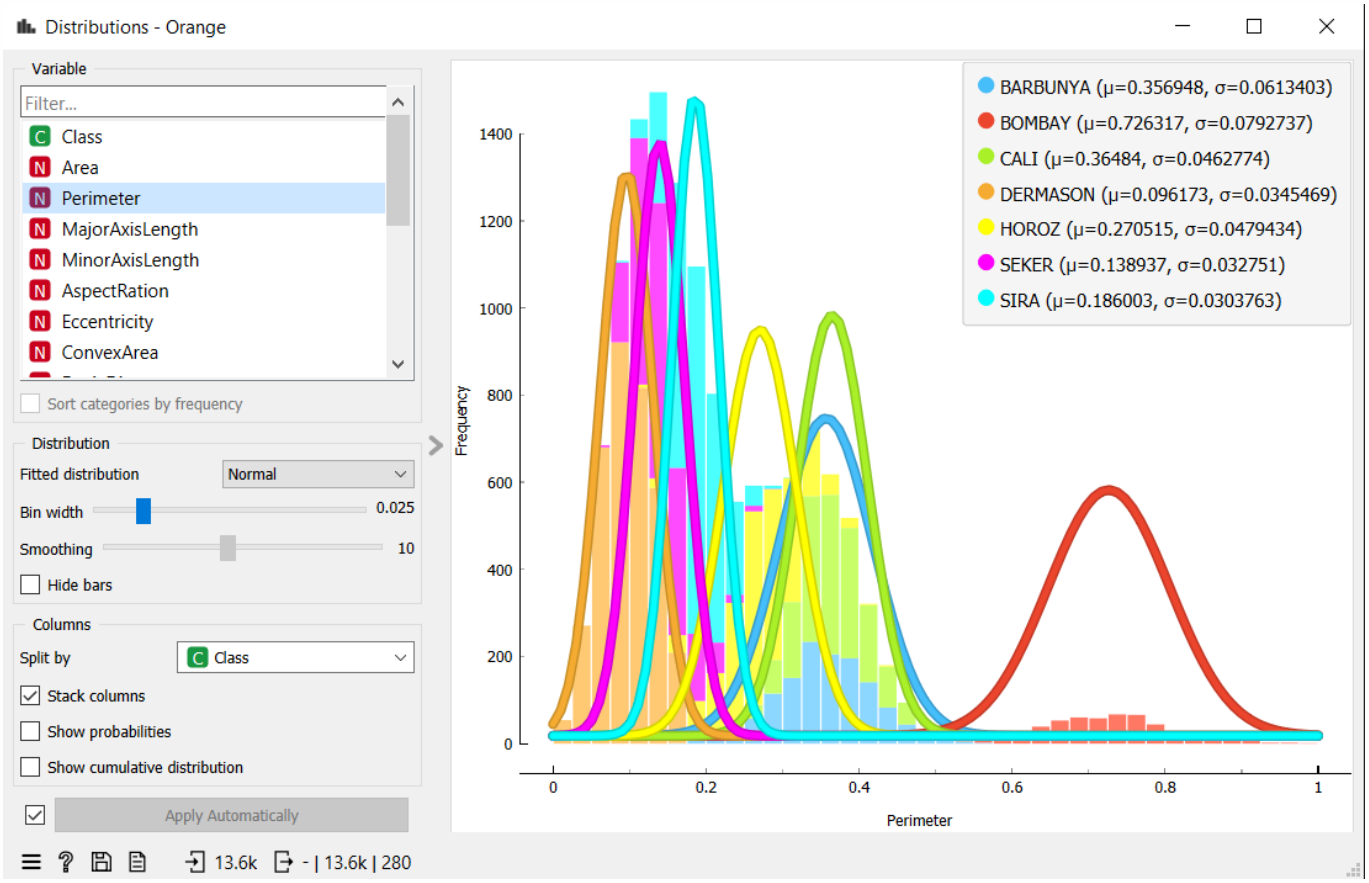
Histogram illustration of distribution of dataset classes



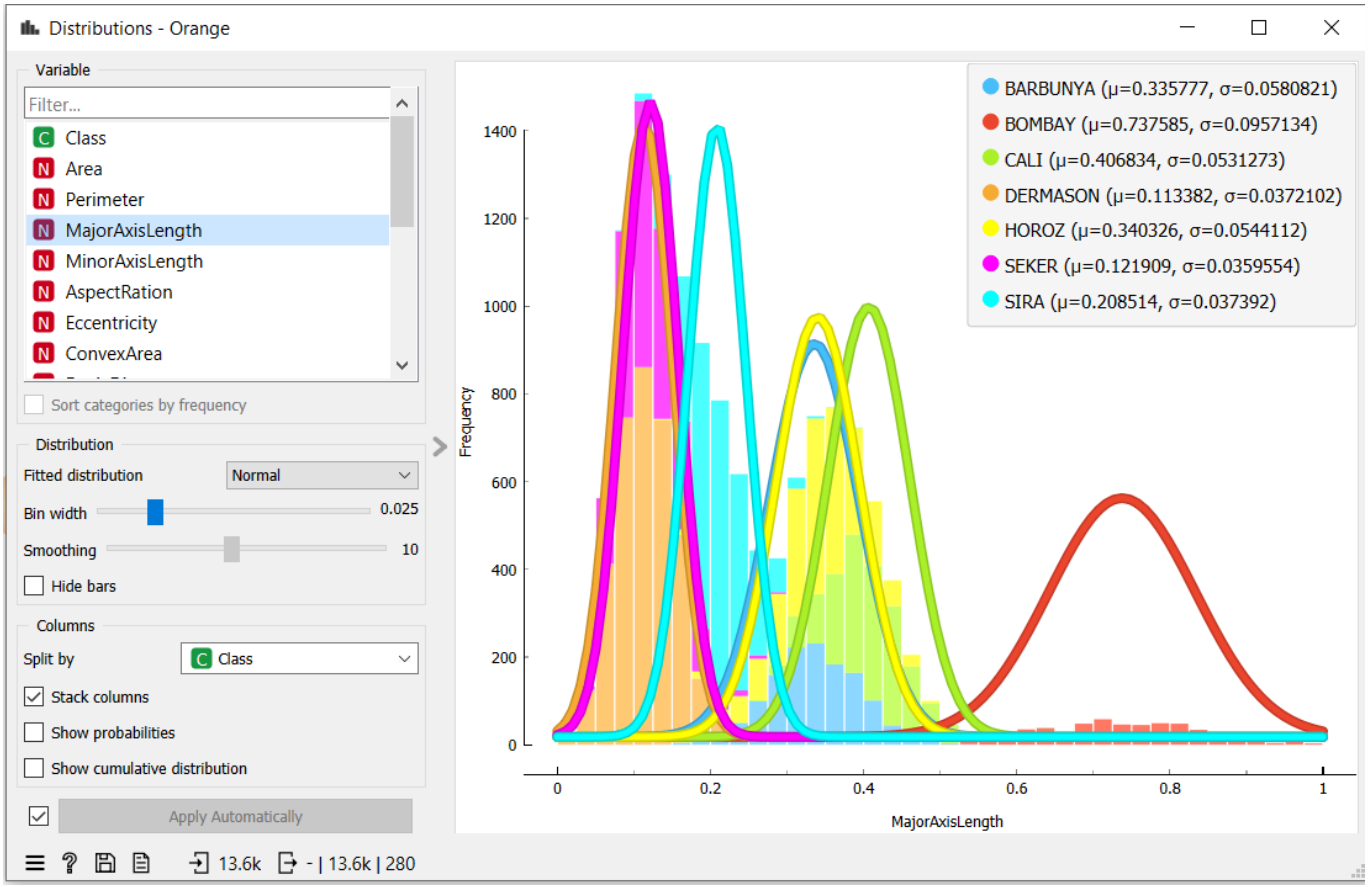
Distribution of classes based on their *compactness*



Distribution of classes based on their *perimeter*



Distribution of classes based on their *Major Axis Length*



Feature Statistic:

Answers to questions

Are the classes in the dataset balanced, or does one class (or several classes) prevail?

Comparing classes in the dataset and looking on the distribution histogram we can say that classes are not perfectly balanced. DERMASON class is a prior class, when CAL is the underdog. This can influence the performance of machine learning models as the model is biased towards the majority class. We need to use oversampling the minority class or undersampling the majority class.

Does the visual representation of the data allow you to see the structure of the data?

Yes. For this we are using distribution and scatter graphs that help to visualise how data is distributed and relationship between attributes accordingly.

How many data groupings can be identified by studying the visual representation of the data?

We have analysed all combinations of attributes provided in Orange tool scatter plot and identified that on each graph representation we can see seven distinct data groupings. Even the relationship between two variables - “Extend” and “Solidity” that is considered as the worst combination of variables to show class separation shows us this separation perfectly. This separation can be not so implicit, but it exists. By taking into consideration the other bad combination, there we can do the same, the separation exists, but it overlaps at the time.

Are the identified data groupings close to each other or far from each other?

Going back to the previous question, we noticed that all classes are separated. But at the same time, we noticed that all classes are placed close to each other except for BOMBAY. It is indicated as a red colour, and it is always placed the farthest away from the rest. We come to the conclusion that this is caused by the unique characteristics of the BOMBAY class. It significantly differentiates from the other classes approximately in all features except for Roundness, Solistity and Shape Factor 4.

Conclusions arising from the analysis of statistical indicators

During the data exploration we were using different tools from Orange and sometimes by python to code to analyse the dataset . This was our first experience in using Orange and find it very powerful and informative for exploring the data. Particularly we have used Widget for preparing our data for analysis such as Continuize that converts our categorical features to continuous one (in our case we did not have any categorical features), Impute - that sets intervals and relates features to this interval (in our case, we have chosen [0,1]). Then, the part of visualisation comes. We used Scatter plots to see the relationship between features that makes separation between classes. We identified that BOMBAY has the most unique values from the rest of the classes. The separation in other plots were quite good, sometimes it overlapped in the plots that have axis of the worst related features. ALso, we have used a Correlation widget that showed us variables with linear relationships. The most strong were couples of [Area, ConvexArea], [Compactness, ShapeFactor3], [EquivDiameter, Perimeter], [EquivDiameter, Area] , [EquivDiameter, ConvexArea], [ConvexArea, Perimeter], [Area, Perimeter] and some more combinations. These high correlations suggest that these variables are not independent of each other and any changes in one variable are likely to result in changes in the other. This information can be useful in predictive modelling, as you might not need to include all of these highly correlated variables in the model. It could also be useful in understanding the characteristics of the beans. For example, beans with a larger area are likely to have a larger convex area as well. We kept our work on by using a Distribution widget where we visually estimated how our classes are distributed based on different features. Thus, we saw the majority class - DEMARSON and the underdog one - CALI. Besides, we estimated how those distributions fitted to the Normal distribution. Nevertheless, it was only the first glance and we need to use an K-means algorithm to check it.

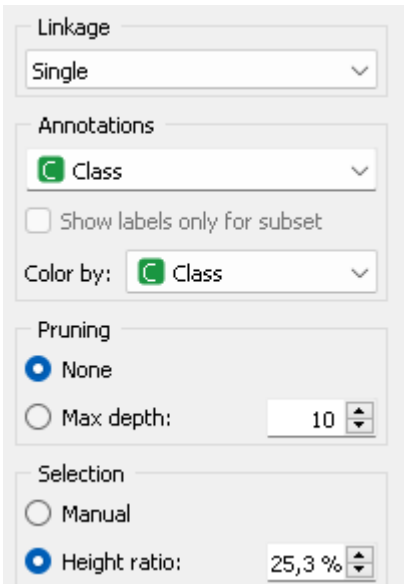
Part II

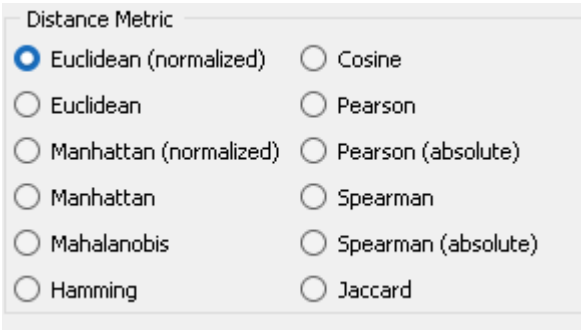
<this subsection should describe the use of unsupervised machine learning algorithms, accompanied by screenshots and references to the information sources used>

Hierarchical clustering

Hyperparameters available in the Orange tool:

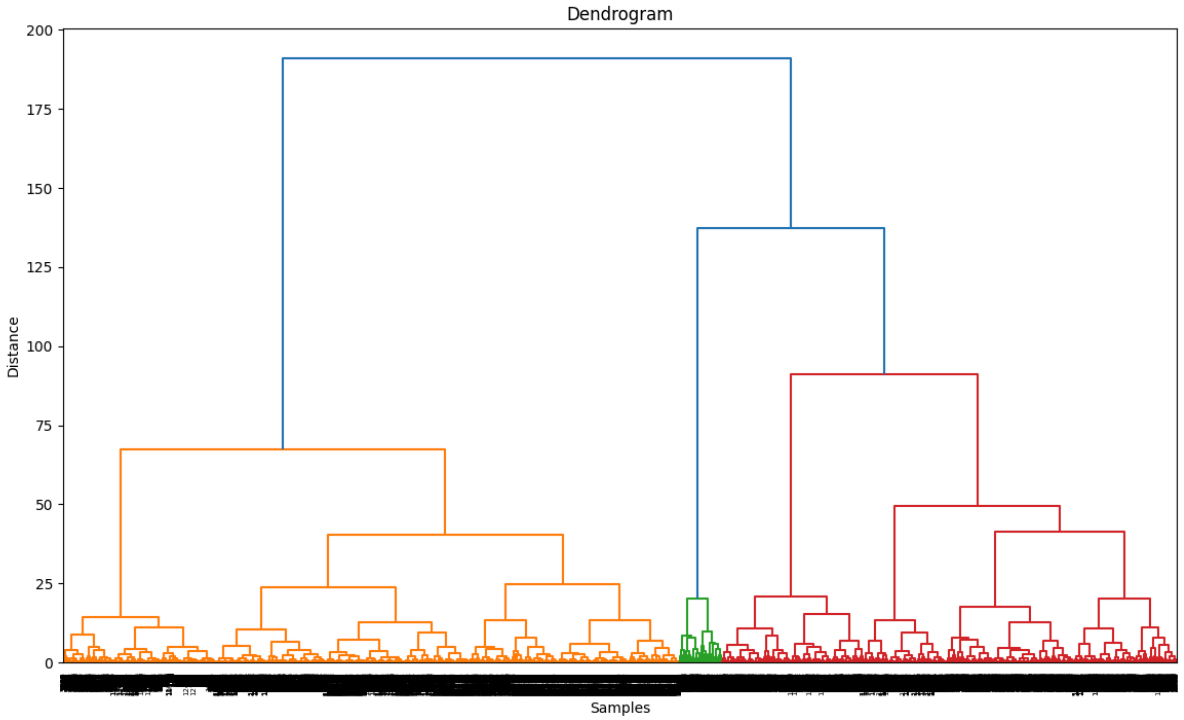
Hyperparameter	Description
Linkage Method	Determines how the distance between clusters is calculated. Common methods include single, complete, average, and Ward's method.
Distance Metric	Defines the metric used to calculate the distance between points in the space. Options can include Euclidean, Manhattan, Cosine, etc.
Pruning	This controls whether and how the dendrogram is pruned. It could be based on the maximum depth of the dendrogram or other criteria.
Height ratio	This could control the colour coding of the dendrogram branches, often based on distance thresholds to visually differentiate clusters.
Annotations	Allows toggling different types of labels and annotations, like class labels, on the dendrogram.
Selection	Can be manual or automated, affecting how clusters are selected or highlighted in the visualisation.



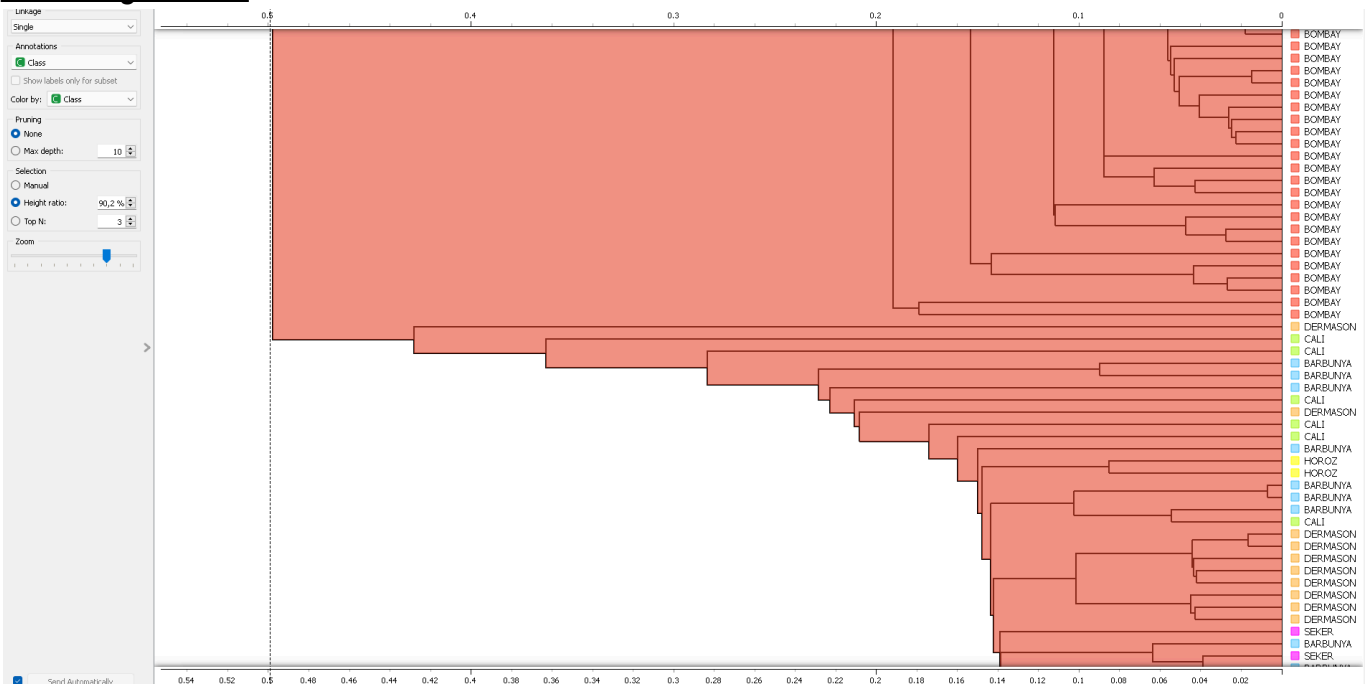


Description of experiments

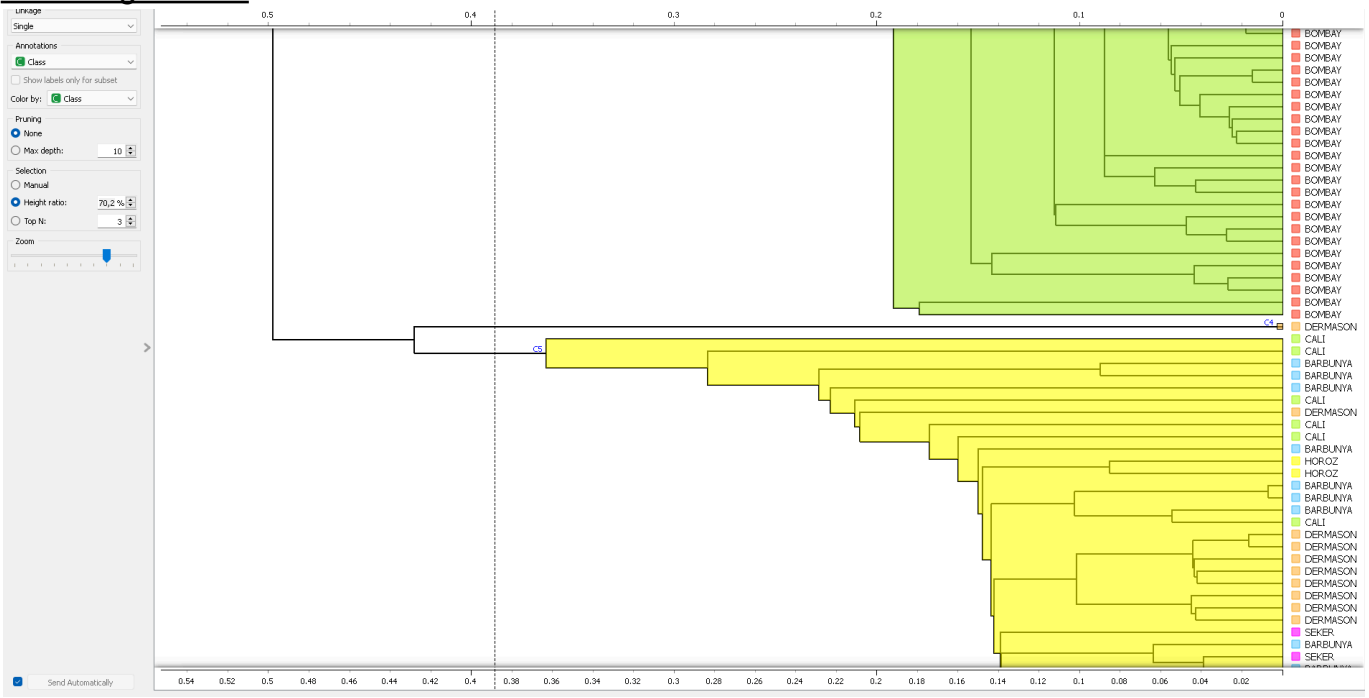
Dendrogram



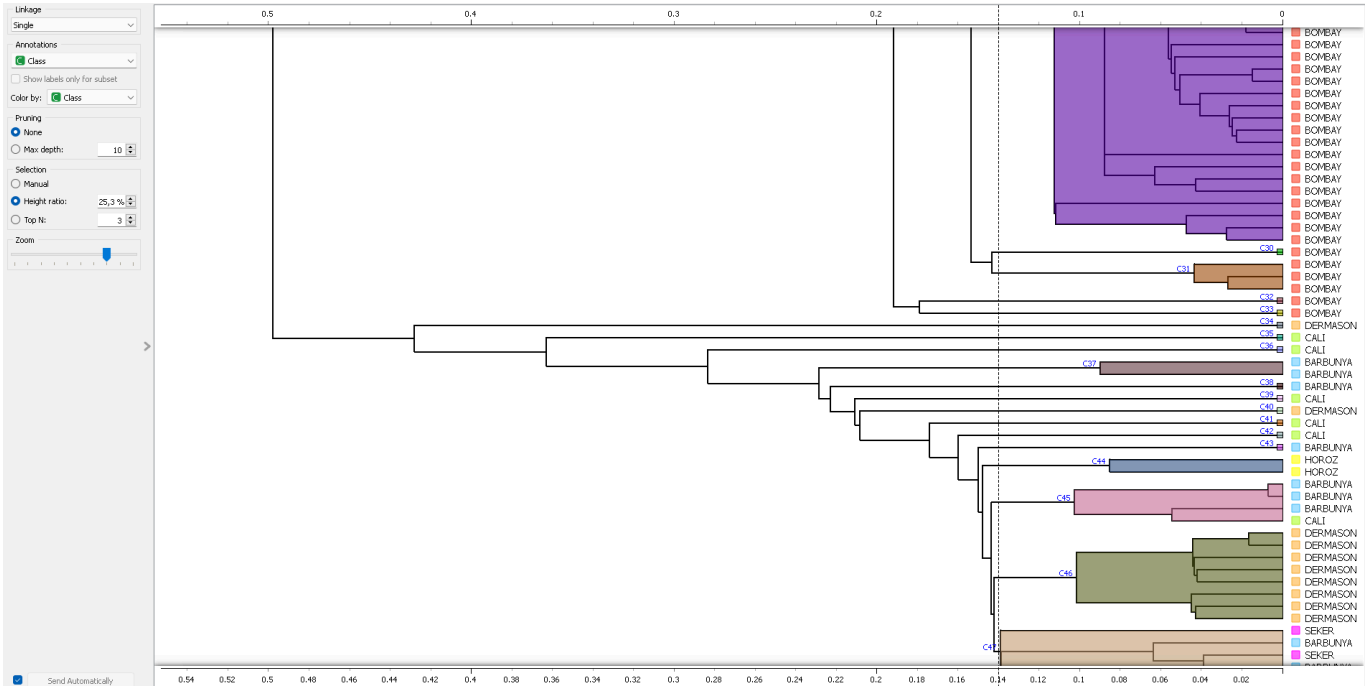
90% Height Ratio



75% Height Ratio



25% Height Ratio



Conclusions from experiments:

- Lowering the cutoff height in hierarchical clustering leads to an increase in the number of clusters and improves class separation.

- At lower cutoffs (like 90%), the clusters tend to be more homogeneous, which could be useful if the goal is to identify tightly-knit groups within the data however too low a cutoff might result in overfitting, where too many small clusters are formed, like a for making the clustering less useful for broader categorization tasks.

It is essential to choose a cutoff level that effectively balances between under-clustering (too few clusters, insufficient detail) and over-clustering (too many clusters, excessive fragmentation).

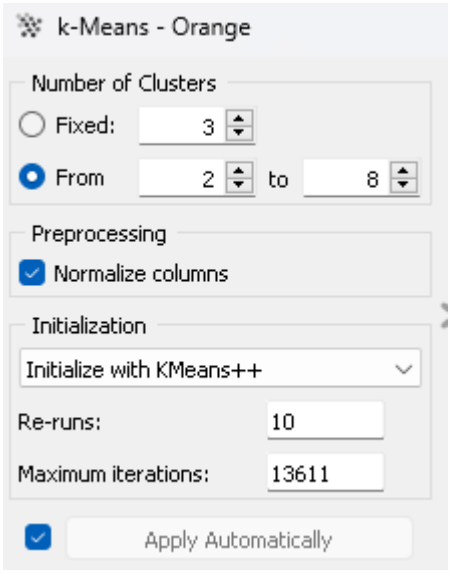
K-means algorithm

Hyperparameters available in the Orange tool:

<add rows to table as needed>

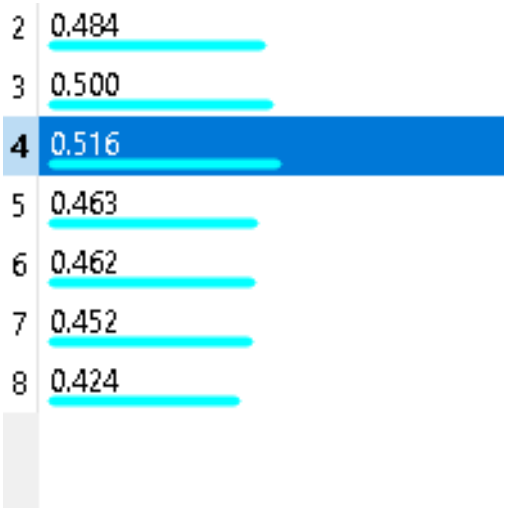
Hyperparameter	Description
Number of Clusters	it defines the number of distinct clusters that the algorithm will form.
Preprocessing:	Normalise columns: A preprocessing step where each column of the dataset is scaled to have a mean of zero and a standard deviation of one. This normalisation ensures that all features contribute equally to the computation of distances between data points.
Initialization	Initialise with KMeans++: A selection for the initialization method that improves the probability of convergence on optimal cluster centroids by strategically positioning the initial centroids before proceeding with the iterative algorithm.
Re-runs	Specifies the number of times the clustering algorithm is executed with different random seeds.The objective is to minimise variability in results due to random initialization and to ensure reliability in the clustering outcomes.
Maximum iterations	This sets a cap on the number of iterations the algorithm will execute for a single run.

<a screenshot with hyperparameter values set for the algorithm>

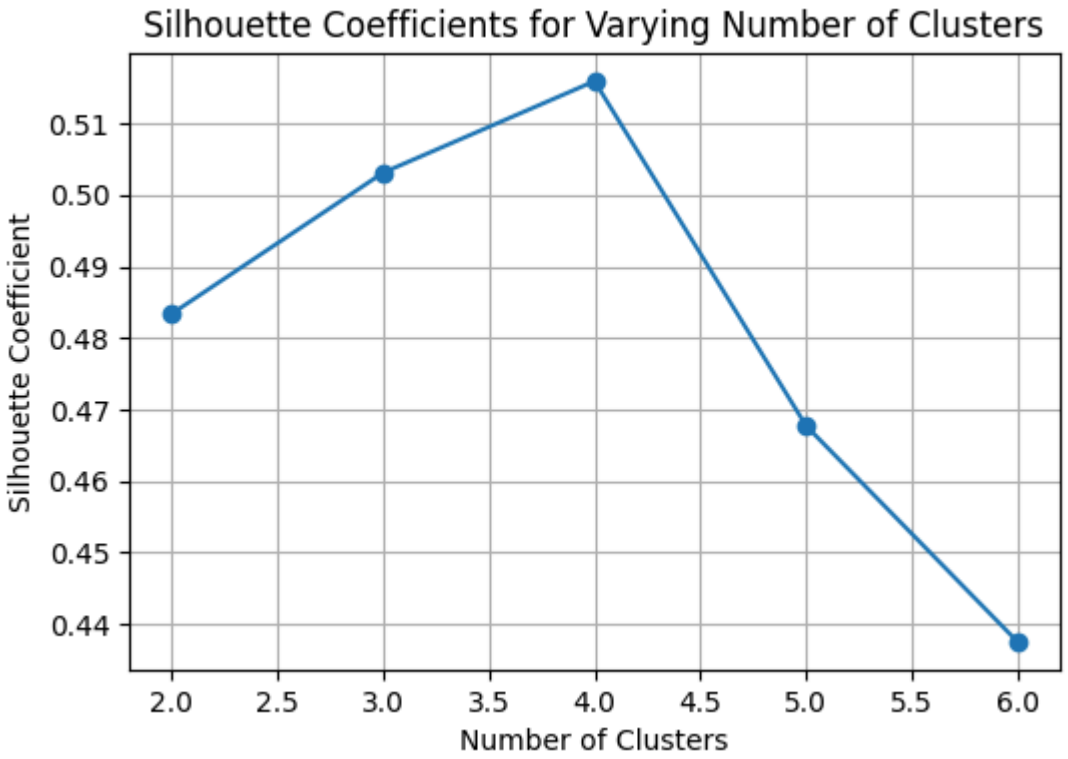


Description of experiments

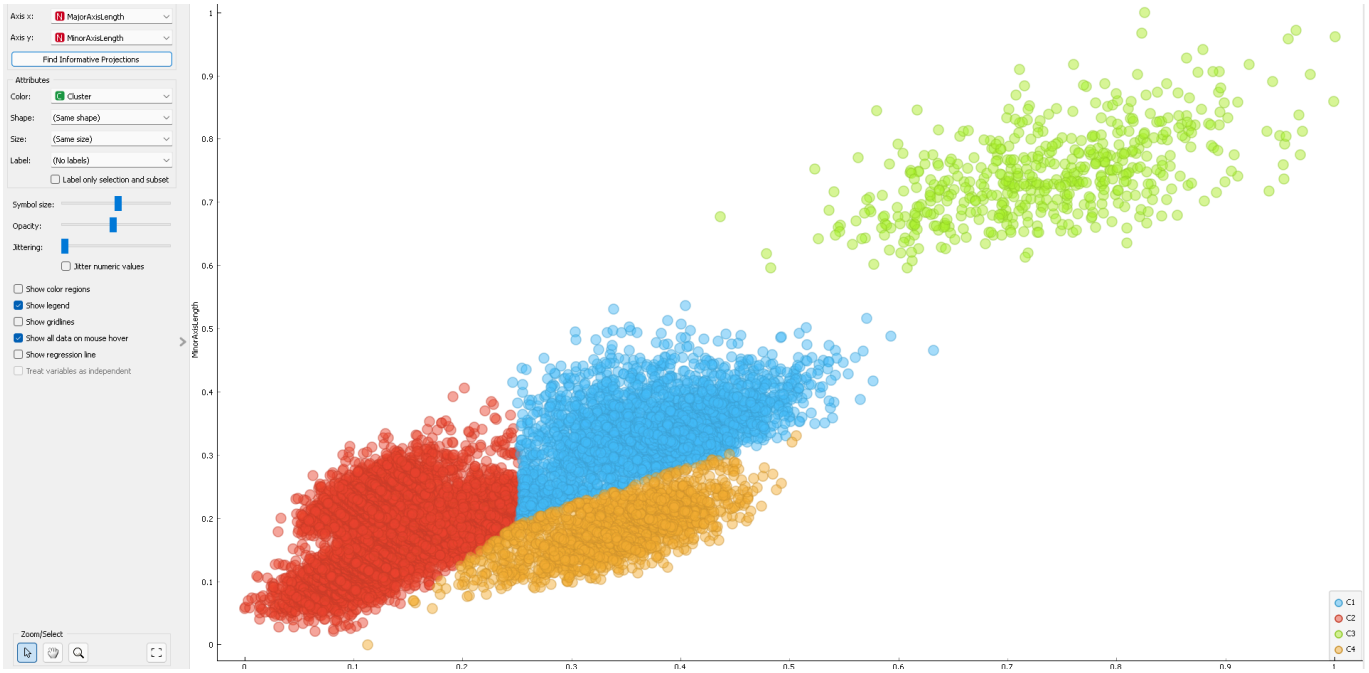
<a screenshot of Silhouette coefficient with at least 5 different k values>




```
Silhouette Score for 2 clusters: 0.483374536450599
Silhouette Score for 3 clusters: 0.5031178547141089
Silhouette Score for 4 clusters: 0.5159289839904073
Silhouette Score for 5 clusters: 0.46774853081548384
Silhouette Score for 6 clusters: 0.4375400349095274
```



<a screenshot of the scatterplot according to the best value of the Silhouette coefficient>



Conclusions from experiments:

The classes in the dataset are indeed separable, particularly at the threshold of four clusters, as supported by both the visual evidence and the silhouette scores and cluster number not only shows the maximum degree of separation but also indicates that the clusters are meaningfully defined, with members more similar to each other than to members of other clusters and the alignment between the highest silhouette score and the visual separability suggests that the clustering method is effectively identifying distinct groups within the data.

Final conclusions

When we used clustering techniques and settings, the dataset's classes were easily separated , both hierarchical and k-means clustering approaches had advantages in demonstrating class separability and the results of the clustering analyses clearly show that the dataset's classes are properly separated.

- K-means clustering produces optimal separation at four clusters, as seen by the greatest silhouette score of around 0.52, indicating distinct and well-defined groupings and hierarchical clustering reinforces this, demonstrating good class separation at lower thresholds where clusters are more homogeneous.

- Hierarchical Clustering is useful for investigating data at different resolutions and comprehending the hierarchical structure of data aggregation and K-means Clustering succeeds at clearly defining and separating groups when the appropriate number of clusters is selected, as evidenced by the high silhouette scores at four clusters.

While working on the k-means algorithm we had some problems with silhouette score and after that decided to create python script to solve this problem and our script was made for clustering analysis on a dataset using the K-Means algorithm, which will evaluates the clustering performance with the Silhouette coefficient, and visualises the results.

Overall, on this part of practical assignment we understood that the classes are well-separable under optimised settings for each clustering method, implying that the dataset is divided into different groups that can be efficiently identified by both hierarchical and K-means clustering methods.

Part III

<this subsection should describe the use of supervised machine learning algorithms, accompanied by screenshots and references to the information sources used>

Description of the selected algorithms

<a description of freely chosen algorithms and the rationale for their choice (except artificial neural network)>

Title of the first algorithm:

kNN(k-nearest neighbours)

Description of the first algorithm:

The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

Title of the second algorithm:

Description of the second algorithm:

Description of hyperparameters

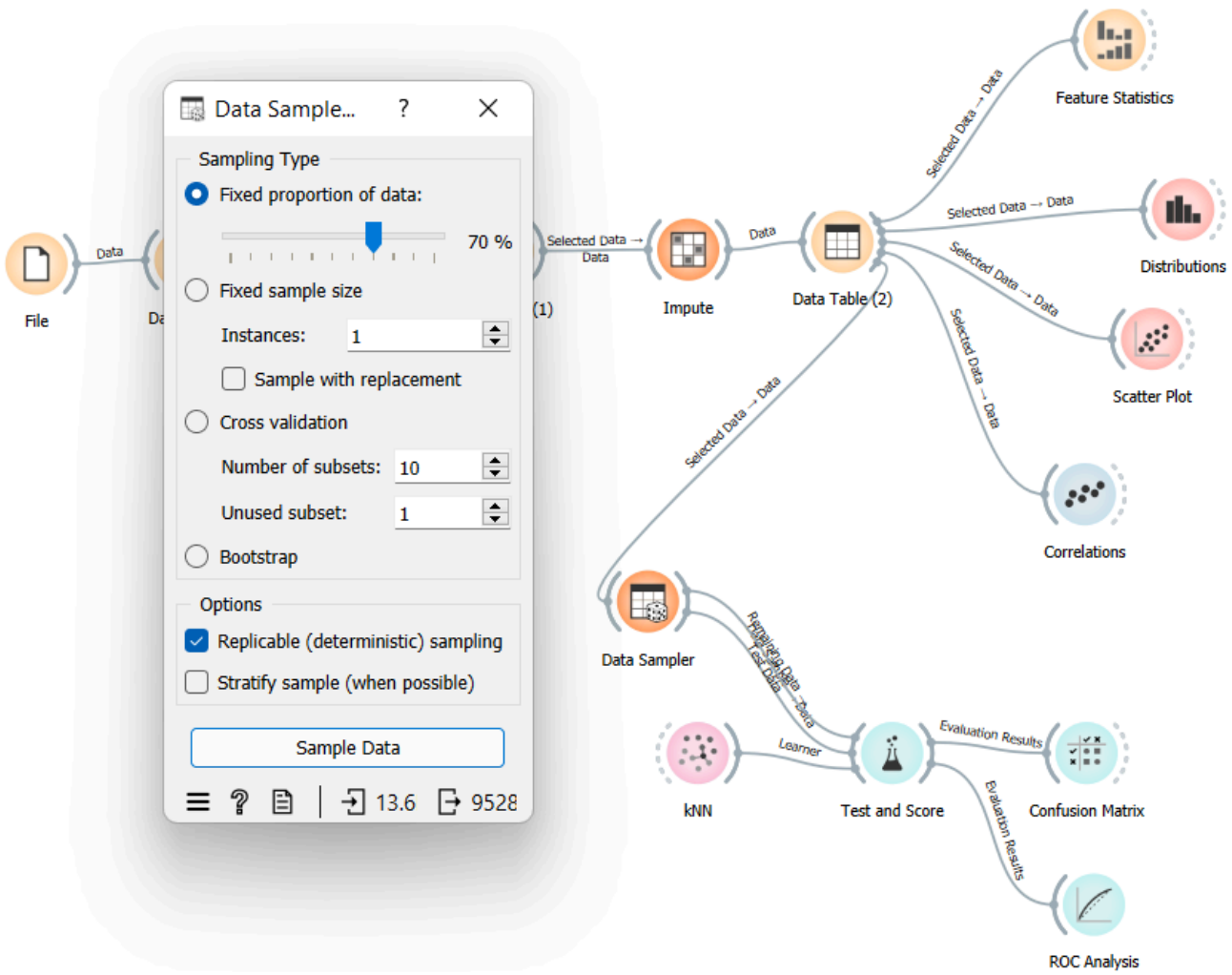
<a description of the hyperparameters available in the Orange tool should be given for each of the algorithms, adding rows to the table as necessary>

Hyperparameter	Description and values
Artificial Neural Networks	
1. Neurons in Hidden Layer	This parameter specifies the architecture of the hidden layer in the network. Each number represents the number of neurons in a respective layer. Values:- 100,100 indicates two hidden layers, each with 100 neurons.
2. Activation	The activation function is used to introduce non-linearity into the output of a neuron. Options:- logistics, identity,tanh, and ReLu. chosen:- Logistic- A sigmoid function that outputs values between 0 and 1. Mostly used for binary classification.

3. Solver	This parameter defines the method to use for optimizing the neural network's weights. Options:- SGD,L-BFGS-B, Adam. Chosen:- SGD (Stochastic Gradient Descent). It is a popular and efficient gradient algorithm, particularly useful for large datasets and parameters.
4. Regularization, α=0.1	Regularization helps to prevent the model from overfitting by penalizing larger weights. The parameter α controls the strength of the regularization. A higher value increases the regularization effect, potentially leading to simpler models.
5. Maximal Number Of Iterations	This sets the limit on the number of iterations the solver will run,which can affect the training time and convergence of the model. Set value - 1000 iterations.
6. Replicable Training	It ensures that the training process can be replicated exactly. It is useful for debugging or for scientific experiments where reproducibility is essential.
kNN	
Number of neighbors	Indicates how many of the closest neighbors should be taken into account when predicting. Values: 1-100
Metric	Gives the option to choose the distance metric that will be used to compare the similarity of the data points. Values: Euclidean, Manhattan, Chebyshev, Mahalanobis
Weight	Establishes the prediction's weighting structure. Values: Uniform, by distances
Logistical Regression	

Information about test and training datasets

<a screenshot of splitting dataset into test and training datasets>



Number of data objects in the training dataset:

9528 instances

% proportion of data objects in the training dataset:

<add rows to table as needed>

Class label	Number of data objects in the training dataset	% proportion of data objects in the training dataset
BARBUNYA	9528	70%
BOMBAY		
CALI		
DERMASON		
HOROZ		
SECER		
SIRA		

Number of data objects in the test dataset:

4083

% proportion of data objects in the test dataset:

<add rows to table as needed>

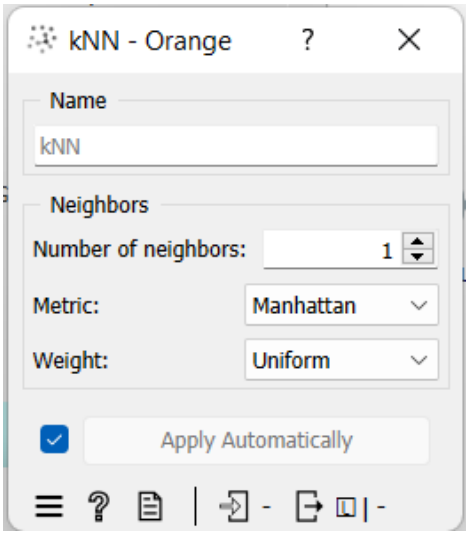
Class label	Number of data objects in the test dataset	% proportion of data objects in the test dataset
BARBUNYA	400	30%
BOMBAY	157	
CALI	489	
DERMASON	1064	
HOROZ	578	
SECER	608	
SIRA	791	

Experiments with kNN (k-nearest neighbours)

<add rows to table as needed>

Experiment	Hyperparameter values
Experiment 1	num of neighbors: 1; metric: Manhattan; weight: uniform
Experiment 2	num of neighbors: 3; metric: Chebyshev; weight: uniform
Experiment 3	num of neighbors: 5; metric: Euclidean; weight: uniform
Experiment 4	num of neighbors: 5; metric: Manhattan; weight: uniform
Experiment 5	num of neighbors: 5; metric: Chebyshev; weight: uniform
Experiment 6	num of neighbors: 5; metric: Euclidean; weight: by distances
Experiment 7	num of neighbors: 5; metric: Chebyshev; weight: by distance

<a screenshot of hyperparameter values for Experiment 1>

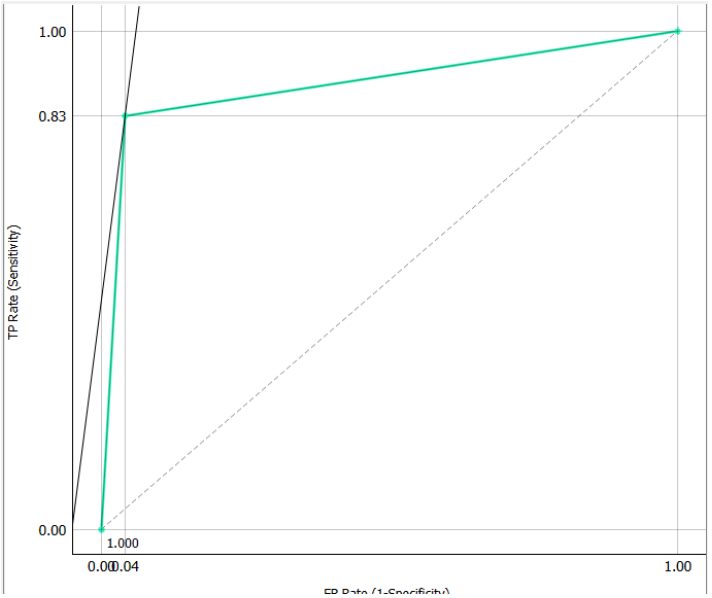


<a screenshot of performance metrics for Experiment 1>

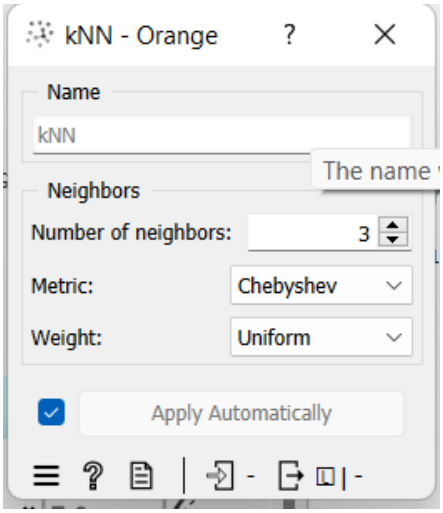
Model	AUC	CA	F1	Prec	Recall	MCC
kNN	0.936	0.894	0.894	0.894	0.894	0.872

Predicted

Actual	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA	Σ
BARBUNYA	814	0	85	1	6	8	23	937
BOMBAY	0	355	0	0	0	0	0	355
CALI	81	0	1062	0	25	2	8	1178
DERMASON	2	0	0	2165	13	52	195	2427
HOROZ	4	0	35	11	1229	0	46	1325
SEKER	6	0	0	47	0	1342	43	1438
SIRA	13	0	8	212	44	41	1550	1868
Σ	920	355	1190	2436	1317	1445	1865	9528



<a screenshot of hyperparameter values for Experiment 2>



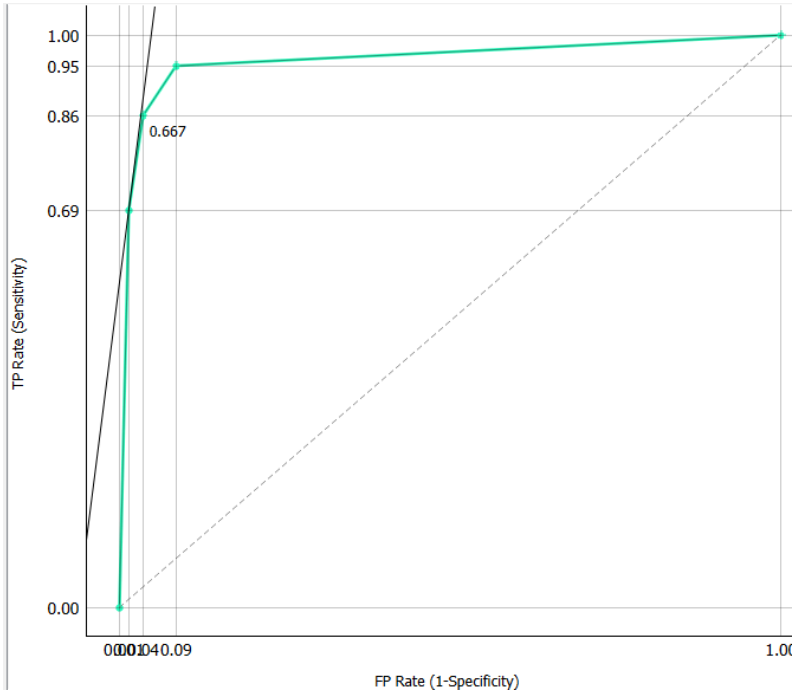
<a screenshot of performance metrics for Experiment 2>

Evaluation results for target (None, show average over classes)

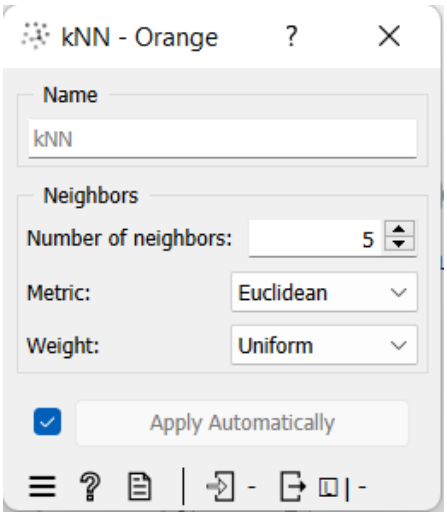
Model	AUC	CA	F1	Prec	Recall	MCC
kNN	0.974	0.913	0.913	0.913	0.913	0.895

Predicted

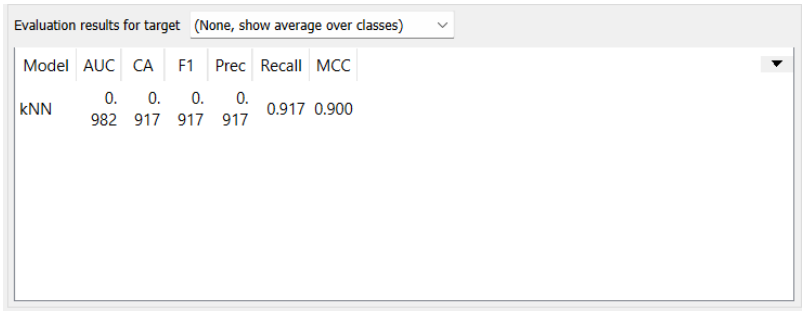
Actual	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA	Σ
BARBUNYA	825.3	0.0	71.7	1.0	6.3	8.0	24.7	937
BOMBAY	0.0	355.0	0.0	0.0	0.0	0.0	0.0	355
CALI	60.7	0.0	1068.7	0.0	31.7	4.0	13.0	1178
DERMASON	1.0	0.0	0.0	2150.0	12.0	45.7	218.3	2427
HOROZ	5.0	0.0	39.7	10.0	1225.7	0.0	44.7	1325
SEKER	11.3	0.0	2.7	47.3	0.0	1331.7	45.0	1438
SIRA	11.7	0.0	11.0	218.3	40.0	30.3	1556.7	1868
Σ	915	355	1194	2427	1316	1420	1902	9528

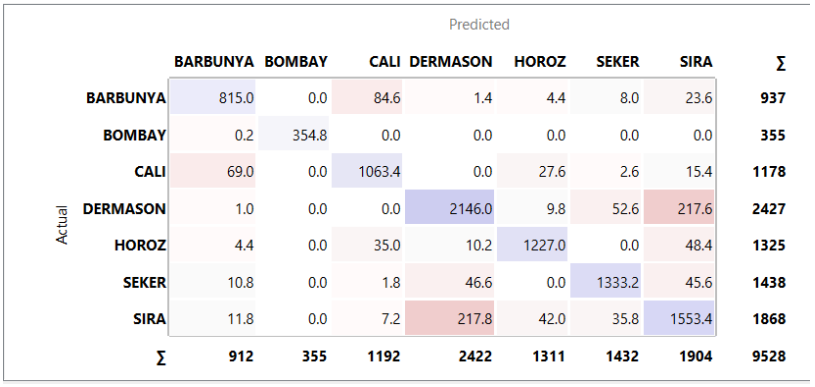


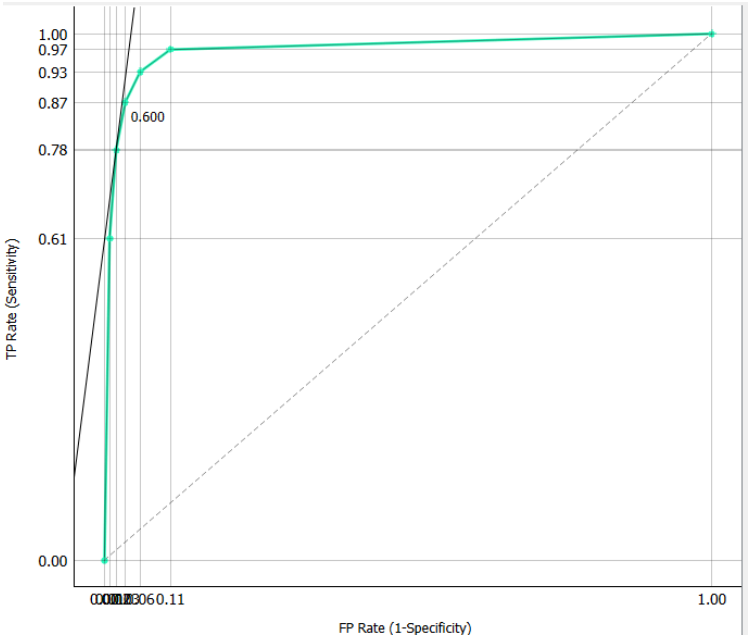
<a screenshot of hyperparameter values for Experiment 3>



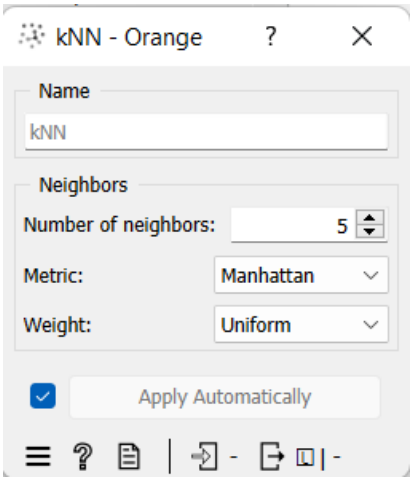
<a screenshot of performance metrics for Experiment 3>



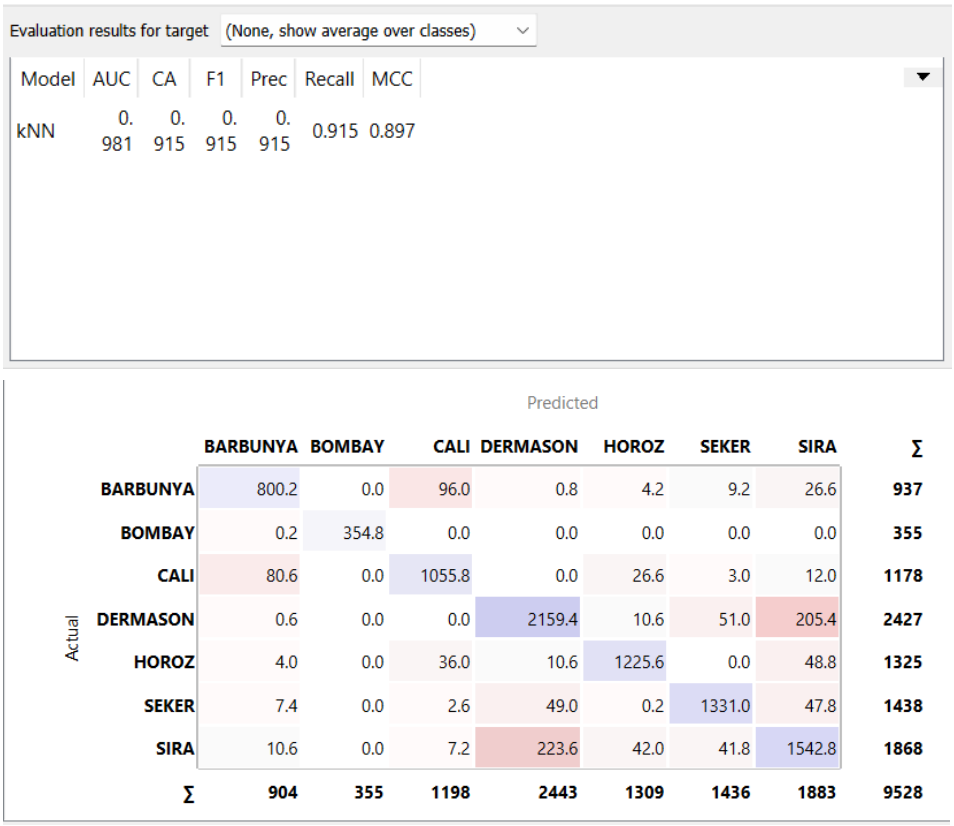


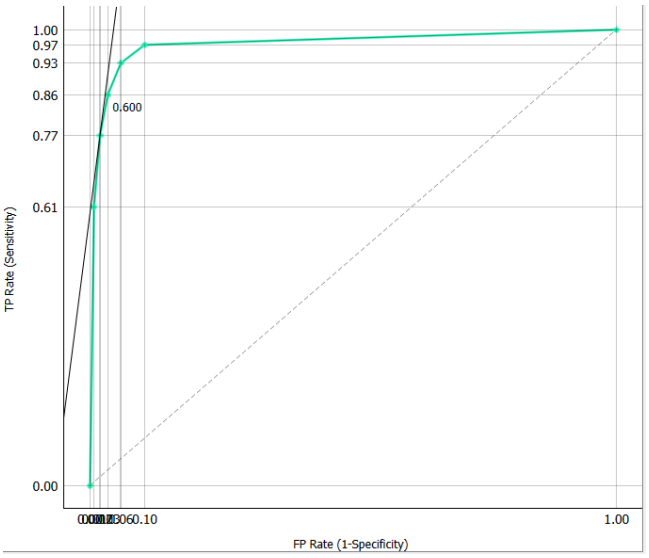


<a screenshot of hyperparameter values for Experiment 4>

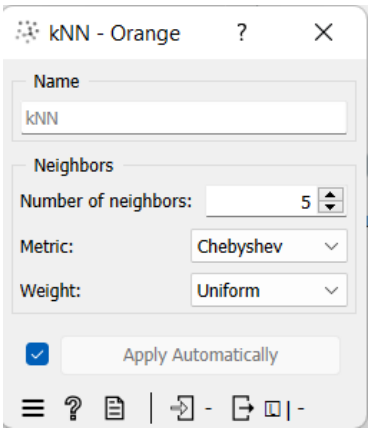


<a screenshot of performance metrics for Experiment 4>

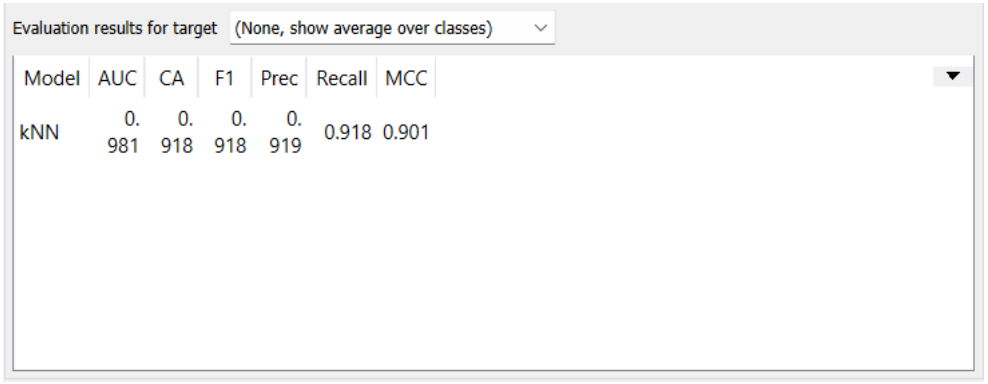




<a screenshot of hyperparameter values for Experiment 5>

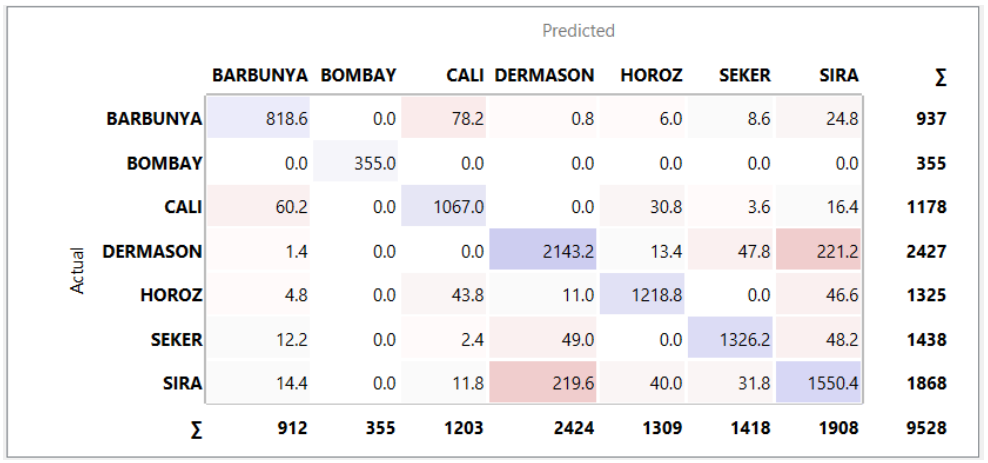


<a screenshot of performance metrics for Experiment 5>



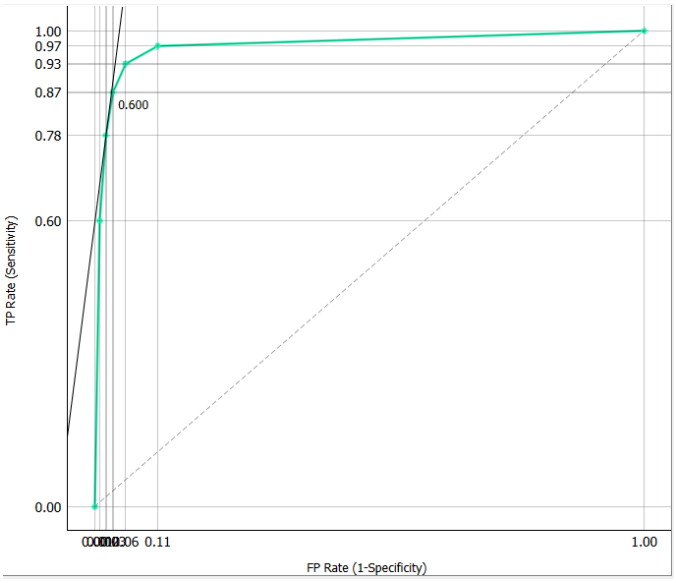
Evaluation results for target (None, show average over classes)

Model	AUC	CA	F1	Prec	Recall	MCC
kNN	0.981	0.918	0.918	0.919	0.918	0.901

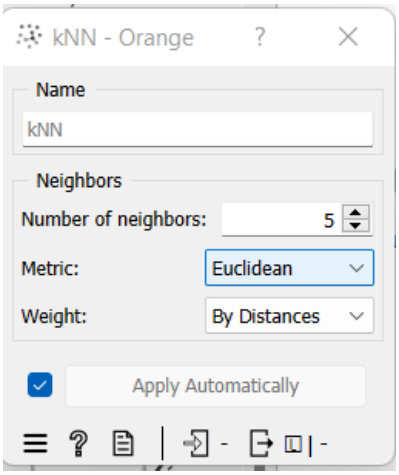


Predicted

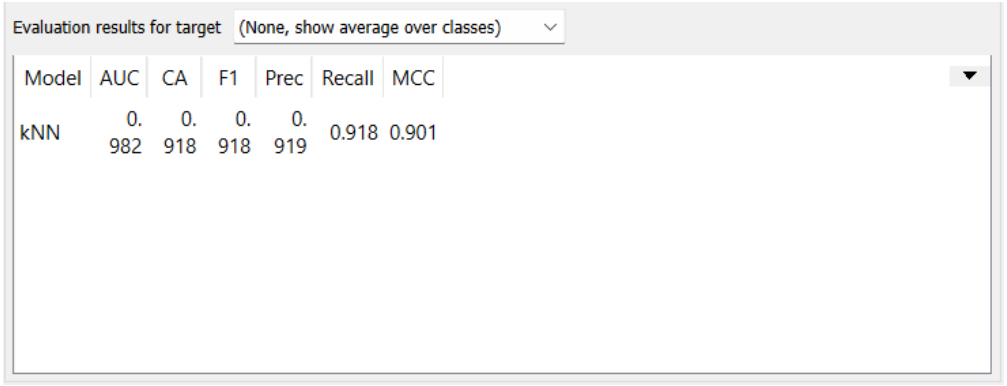
Actual	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA	Σ
BARBUNYA	818.6	0.0	78.2	0.8	6.0	8.6	24.8	937
BOMBAY	0.0	355.0	0.0	0.0	0.0	0.0	0.0	355
CALI	60.2	0.0	1067.0	0.0	30.8	3.6	16.4	1178
DERMASON	1.4	0.0	0.0	2143.2	13.4	47.8	221.2	2427
HOROZ	4.8	0.0	43.8	11.0	1218.8	0.0	46.6	1325
SEKER	12.2	0.0	2.4	49.0	0.0	1326.2	48.2	1438
SIRA	14.4	0.0	11.8	219.6	40.0	31.8	1550.4	1868
Σ	912	355	1203	2424	1309	1418	1908	9528

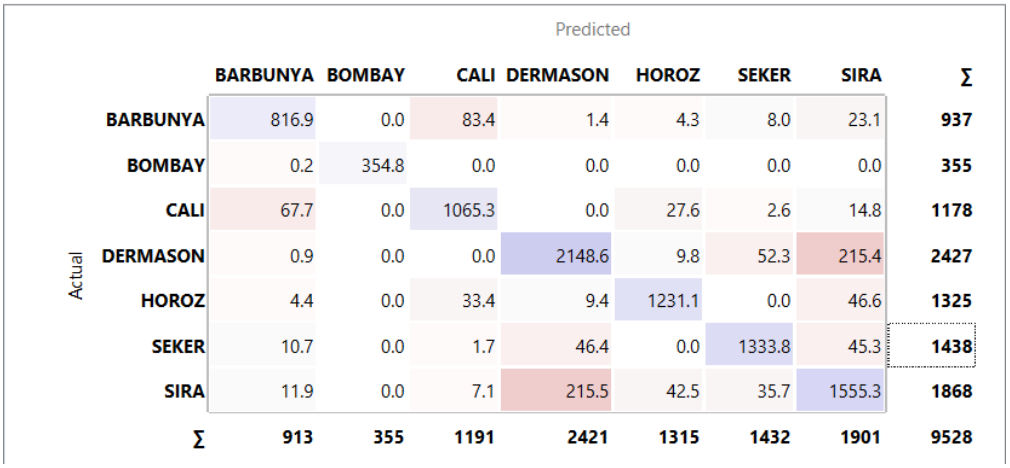


<a screenshot of hyperparameter values for Experiment 6>



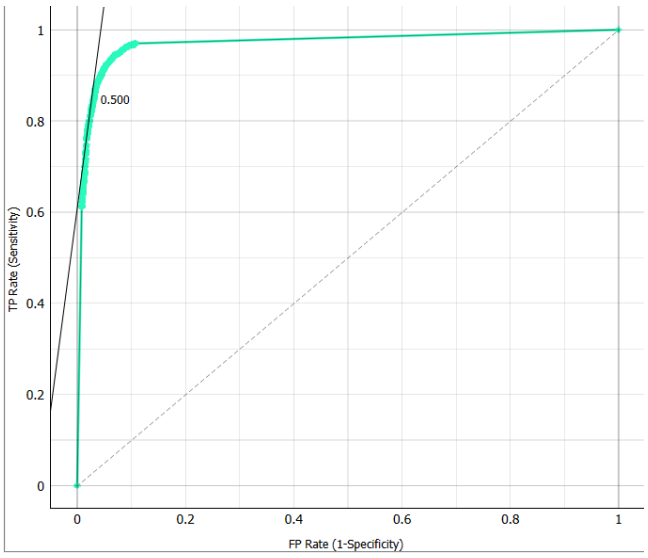
<a screenshot of performance metrics for Experiment 6>



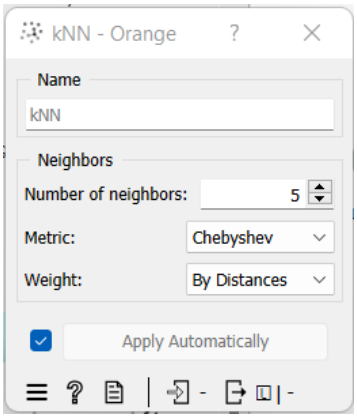


Predicted

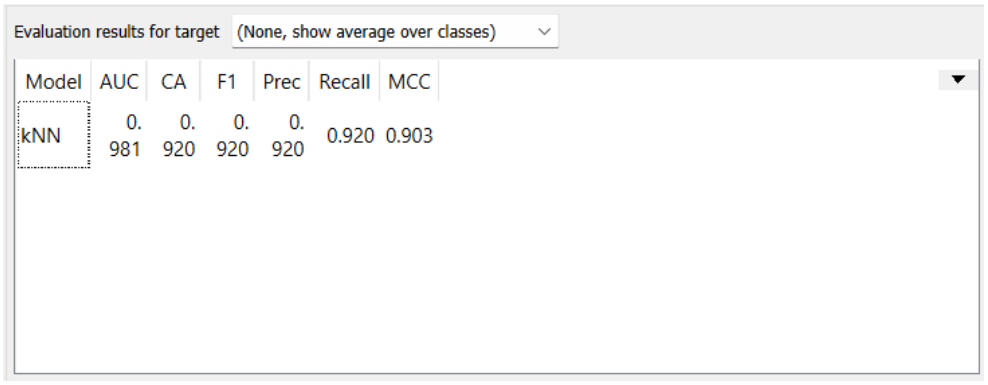
Actual	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA	Σ
BARBUNYA	816.9	0.0	83.4	1.4	4.3	8.0	23.1	937
BOMBAY	0.2	354.8	0.0	0.0	0.0	0.0	0.0	355
CALI	67.7	0.0	1065.3	0.0	27.6	2.6	14.8	1178
DERMASON	0.9	0.0	0.0	2148.6	9.8	52.3	215.4	2427
HOROZ	4.4	0.0	33.4	9.4	1231.1	0.0	46.6	1325
SEKER	10.7	0.0	1.7	46.4	0.0	1333.8	45.3	1438
SIRA	11.9	0.0	7.1	215.5	42.5	35.7	1555.3	1868
Σ	913	355	1191	2421	1315	1432	1901	9528

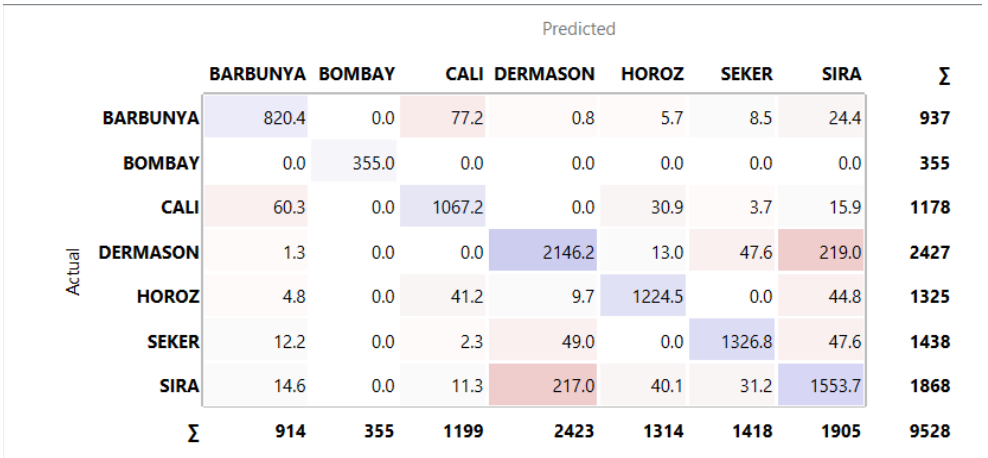


<a screenshot of hyperparameter values for Experiment 7>



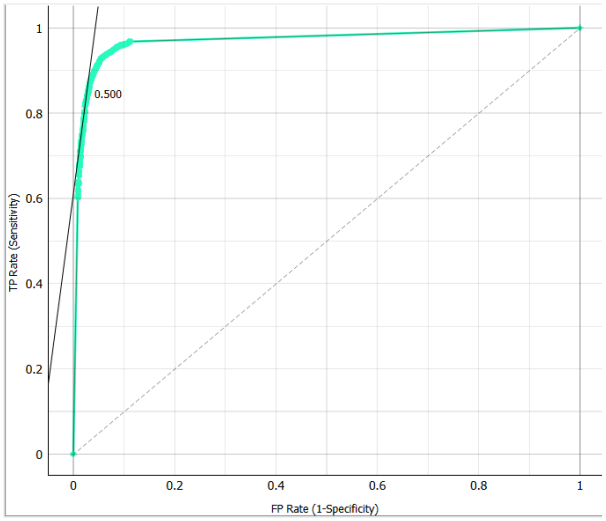
<a screenshot of performance metrics for Experiment 7>





Predicted

Actual	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA	Σ
BARBUNYA	820.4	0.0	77.2	0.8	5.7	8.5	24.4	937
BOMBAY	0.0	355.0	0.0	0.0	0.0	0.0	0.0	355
CALI	60.3	0.0	1067.2	0.0	30.9	3.7	15.9	1178
DERMASON	1.3	0.0	0.0	2146.2	13.0	47.6	219.0	2427
HOROZ	4.8	0.0	41.2	9.7	1224.5	0.0	44.8	1325
SEKER	12.2	0.0	2.3	49.0	0.0	1326.8	47.6	1438
SIRA	14.6	0.0	11.3	217.0	40.1	31.2	1553.7	1868
Σ	914	355	1199	2423	1314	1418	1905	9528



Conclusions from experiments:

<conclusions about the performance of the models in the conducted experiments referring to and analyzing the screenshots above>

A very low value for K such as K=1 or K=3, can be noisy and lead to the effects of outliers in the model. Large values for K are good, but it may find some difficulties. So, the best value for K is 5.

Euclidean distance: This is the most commonly used distance measure, and it is limited to real-valued vectors. It measures a straight line between the query point and the other point being measured.

Manhattan distance: Measures the absolute value between two points.

Chebyshev distance: The determination of the absolute magnitude of the differences between coordinates of a pair of data objects X and Y.

To select the best model we need to look at the results and find the model with hyperparameter values with results closer to 1.

Model selected for testing:

<indication of which experimental model is selected for the testing process>

num of neighbors: 5; metric: Chebyshev; weight: uniform

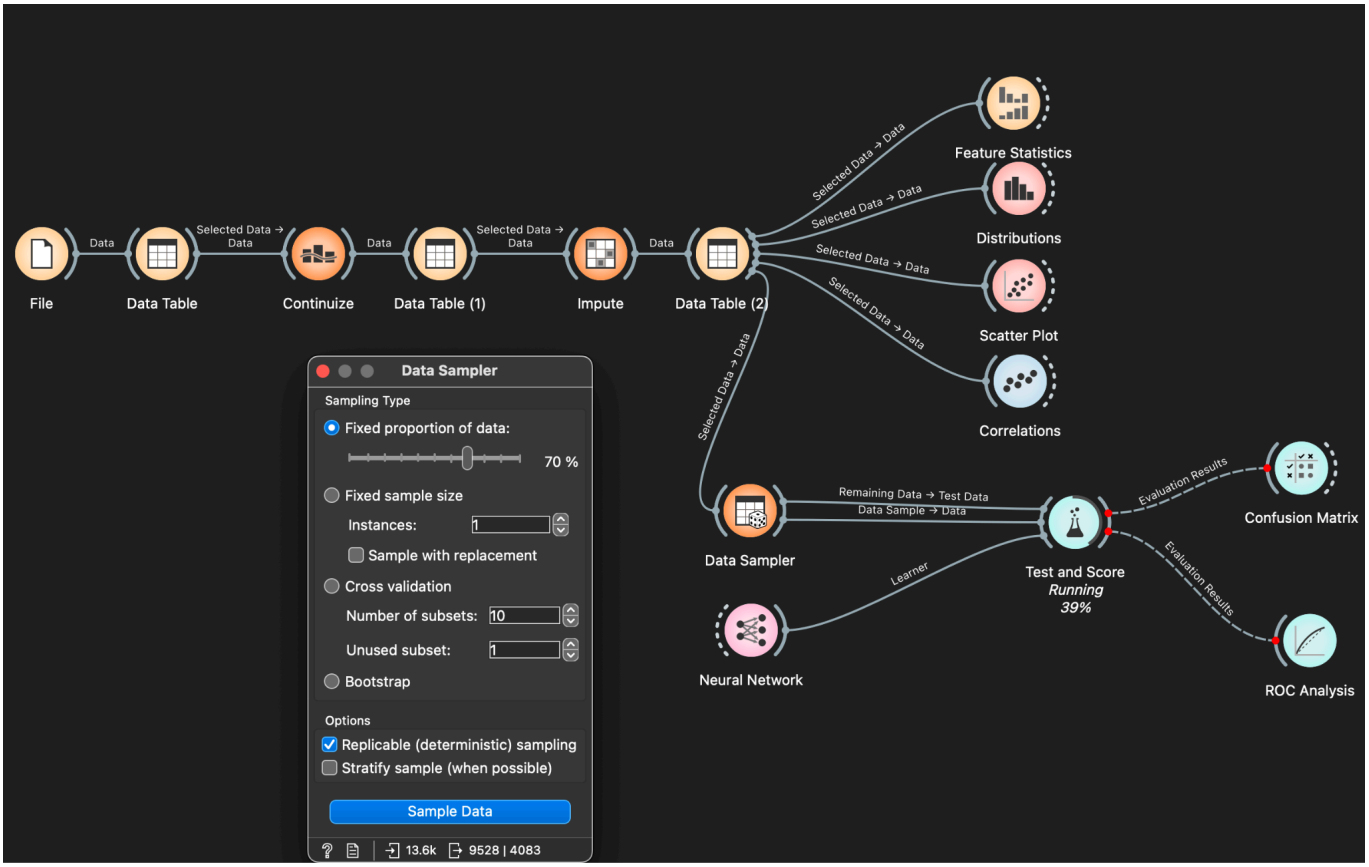
This model has the best results for both uniform and by-distance metrics. But with uniform weighting, all neighbors within the specified number of nearest neighbors contribute equally to the classification decision. This approach treats all neighbors equally, regardless of their distance from the query point.

Description of hyperparameters

Hyperparameter	Description and values
Artificial Neural Networks	
1. Neurons in Hidden Layer	This parameter specifies the architecture of the hidden layer in the network. Each number represents the number of neurons in a respective layer. Values:- 100,100 indicates two hidden layers , each with 100 neurons.
2. Activation	The activation function is used to introduce non-linearity into the output of a neuron. Options:- logistics , identity,tanh, and ReLu. chosen:- Logistic- A sigmoid function that outputs values between 0 and 1. Mostly used for binary classification.
3. Solver	This parameter defines the method to use for optimizing the neural network's weights. Options:- SGD,L-BFGS-B, Adam.

	Chosen:- SGD (Stochastic Gradient Descent). It is a popular and efficient gradient algorithm, particularly useful for large datasets and parameters.
4. Regularization, α =0.1	Regularization helps to prevent the model from overfitting by penalizing larger weights. The parameter α controls the strength of the regularization. A higher value increases the regularization effect, potentially leading to simpler models.
5. Maximal Number Of Iterations	This sets the limit on the number of iterations the solver will run,which can affect the training time and convergence of the model. Set value - 1000 iterations.
6.Replicable Training	It ensures that the training process can be replicated exactly. It is useful for debugging or for scientific experiments where reproducibility is essential.

Information about test and training datasets



Number of data objects in the training dataset:

9528 instances

% proportion of data objects in the training dataset:

Class label	Number of data objects in the training dataset	% proportion of data objects in the training dataset
BARBUNYA	9528	70%
BOMBAY		
CALI		
DERMASON		
HOROZ		
SECER		
SIRA		

Number of data objects in the test dataset:

4083

% proportion of data objects in the test dataset:

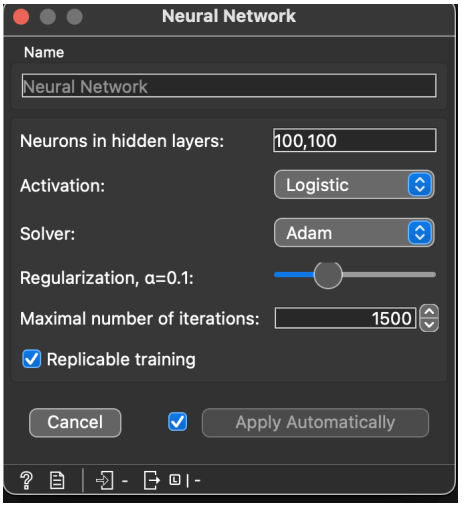
Class label	Number of data objects in the test dataset	% proportion of data objects in the test dataset
BARBUNYA	4083	30%
BOMBAY		
CALI		
DERMASON		
HOROZ		
SECER		
SIRA		

Experiments with Artificial Neural Network

Experiment	Hyperparameter values
Experiment 1	Neuron's :- 100,100 ; Activation:- Logistics ; Solver:- Adam; Regularization:- 0.1; Maximal Number Of Iterations:-1500 ; Replicable Training:- Yes
Experiment 2	Neuron's :- 100,100 ; Activation:- ReLu ; Solver:- Adam; Regularization:- 0.1 ; Maximal Number Of Iterations:- 1500 ; Replicable Training:- Yes

Experiment 3	Neuron's :- 90,90 ; Activation:- Tanh ; Solver:- SGD ; Regularization:- 0.1; Maximal Number Of Iterations:- 1000; Replicable Training:- Yes

<a screenshot of hyperparameter values for Experiment 1>

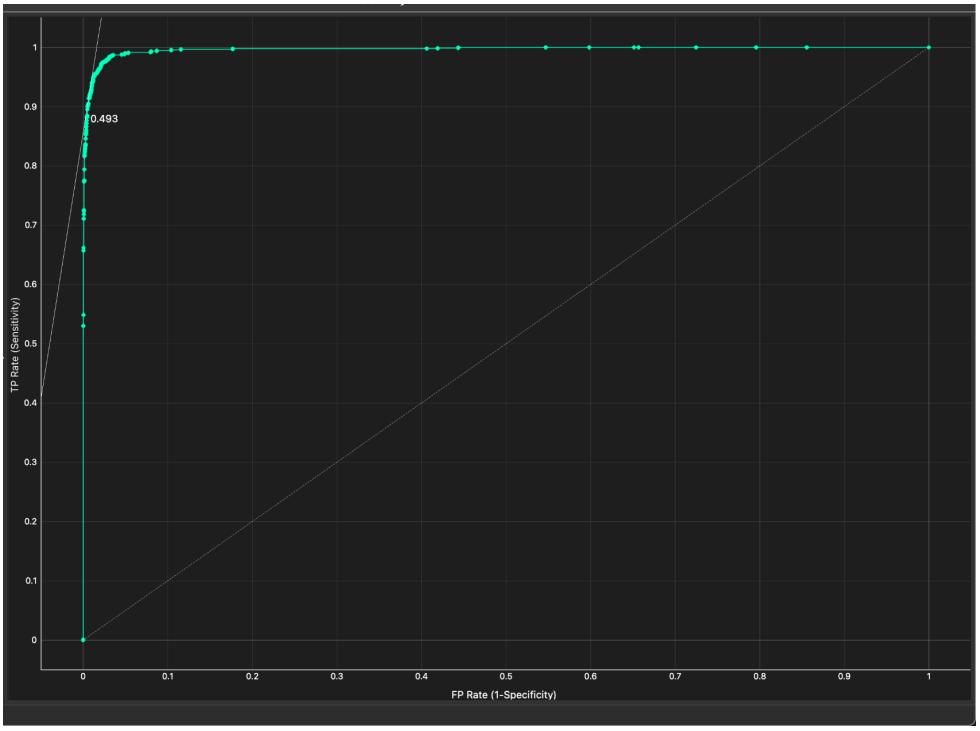


<a screenshot of performance metrics for Experiment 1>

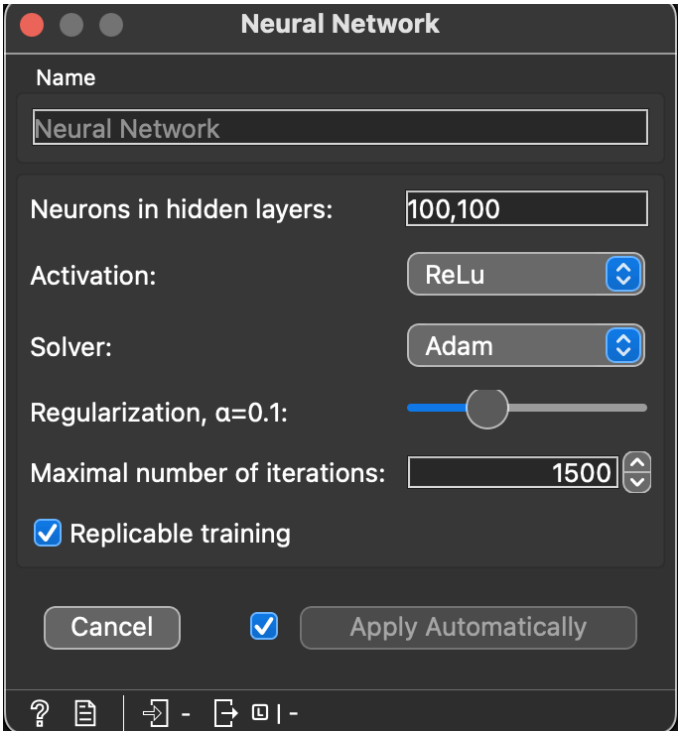
Model	AUC	CA	F1	Prec	Recall	MCC
Neural Network	0.994	0.925	0.925	0.925	0.925	0.909

Predicted

Actual	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA	Σ
BARBUNYA	814.1	5.4	72.8	0.4	4.0	13.9	26.4	937
BOMBAY	4.4	341.7	8.5	0.0	0.0	0.4	0.0	355
CALI	73.0	9.2	1036.5	0.1	35.2	3.8	20.1	1178
DERMASON	0.7	0.0	0.1	2149.0	11.0	51.9	214.2	2427
HOROZ	6.6	0.1	38.2	15.4	1215.4	0.1	49.3	1325
SEKER	15.8	0.1	1.4	55.0	0.1	1314.0	51.5	1438
SIRA	17.4	0.0	20.7	244.9	61.1	50.7	1473.2	1868
Σ	932	357	1178	2465	1327	1435	1835	9528



<a screenshot of hyperparameter values for Experiment 2>

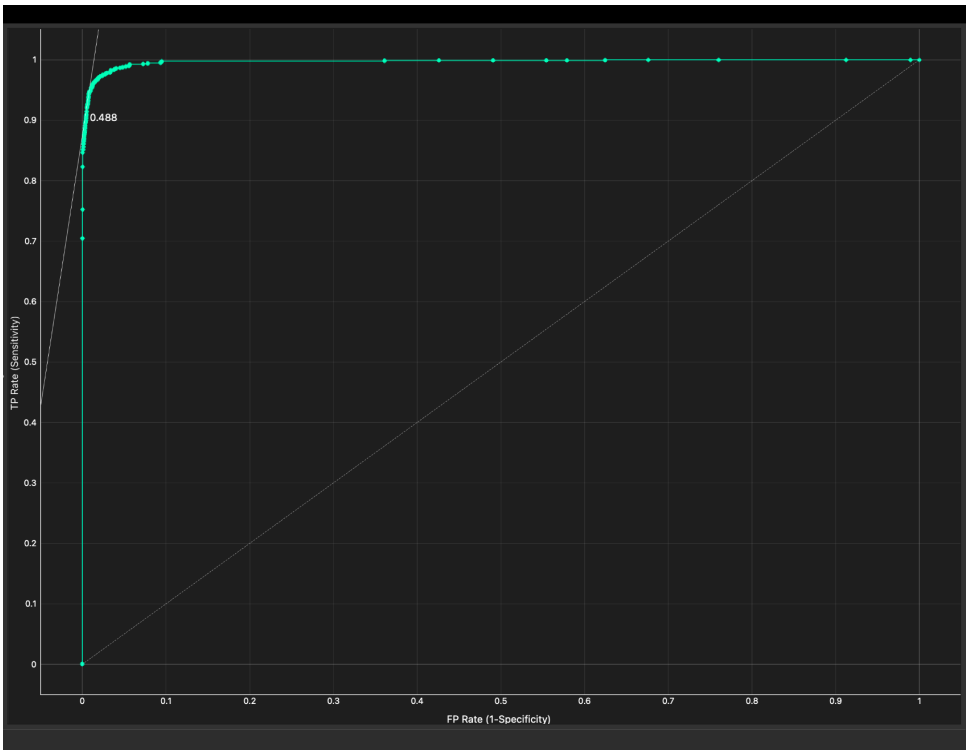


<a screenshot of performance metrics for Experiment 2>

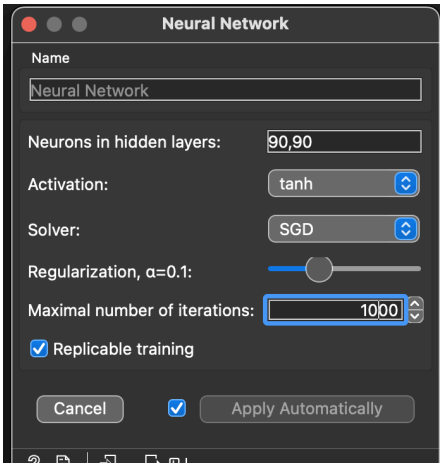
Model	AUC	CA	F1	Prec	Recall	MCC
Neural Network	0.995	0.932	0.932	0.932	0.932	0.917

Predicted

Actual	BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA	Σ
BARBUNYA	848.8	0.5	58.9	1.0	5.5	7.8	14.5	937
BOMBAY	0.3	353.6	1.1	0.0	0.0	0.0	0.0	355
CALI	49.2	1.3	1079.5	0.0	33.1	4.4	10.5	1178
DERMASON	0.8	0.1	0.2	2187.7	12.4	46.3	179.4	2427
HOROZ	4.8	0.0	26.5	14.1	1246.9	0.2	32.6	1325
SEKER	12.1	0.2	3.3	39.2	0.2	1345.9	37.1	1438
SIRA	13.9	0.1	11.9	213.9	50.7	39.6	1537.9	1868
Σ	930	356	1181	2456	1349	1444	1812	9528



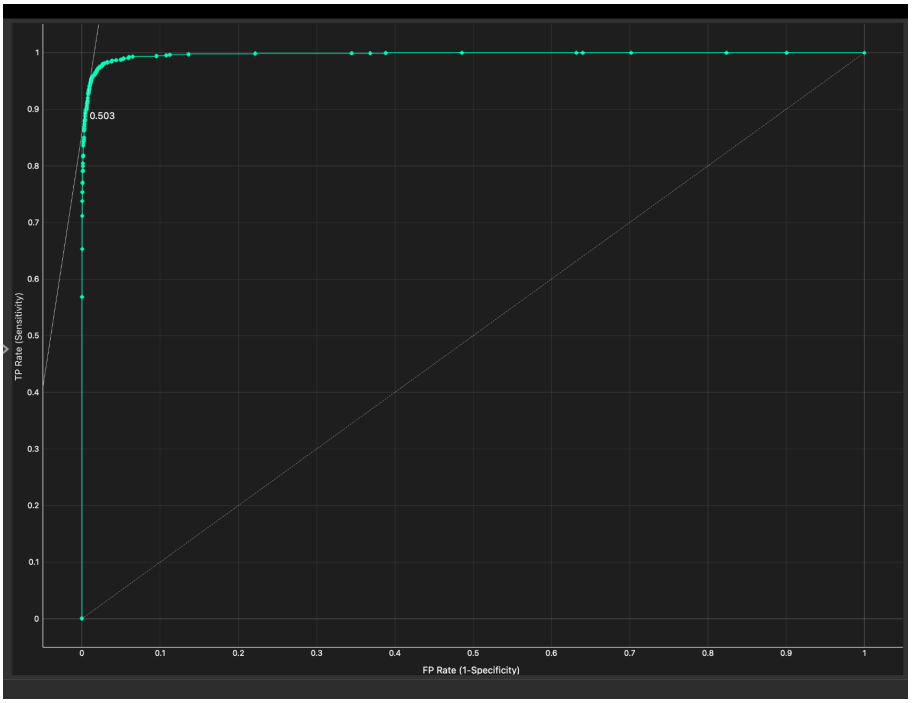
<a screenshot of hyperparameter values for Experiment 3>



<a screenshot of performance metrics for Experiment 3>

Model	AUC	CA	F1	Prec	Recall	MCC
Neural Network	0.994	0.926	0.926	0.926	0.926	0.911

		Predicted							
		BARBUNYA	BOMBAY	CALI	DERMASON	HOROZ	SEKER	SIRA	Σ
Actual	BARBUNYA	835.1	1.5	65.7	0.4	3.5	8.6	22.2	937
	BOMBAY	1.6	349.6	3.6	0.0	0.0	0.2	0.0	355
	CALI	67.6	3.6	1057.9	0.0	29.3	3.0	16.5	1178
	DERMASON	1.1	0.0	0.1	2150.4	8.1	47.5	219.8	2427
	HOROZ	6.8	0.0	32.7	13.3	1226.7	0.0	45.3	1325
	SEKER	15.9	0.7	1.6	45.3	0.0	1327.1	47.5	1438
	SIRA	13.3	0.1	17.2	213.4	52.3	46.1	1525.6	1868
	Σ	941	356	1179	2423	1320	1432	1877	9528



Conclusions from experiments:

Experiment 1:- Activation:- Logistics and Solver:- Adam

1. Confusion Matrix Overview- It shows values for multiple classes (BARBUNYA,BOMBAY, CALI,etc.) Classes like DERMASON show a high number of correct predictions - about 2149 correct—which suggests good model performance for these classes. However,there are also noticeable misclassifications - about 1215 for HOROZ to SEKER—indicating areas where the model struggles/class boundaries might be overlapping or not well defined by the features.

2. ROC Curve Analysis- For the BARBUNYA class, ROC indicates an excellent true positive rate (TPR) across most threshold settings, with a rapid climb towards a high TPR at low false positive rates (FPR). The area under the curve (AUC) for BARBUNYA is close to 1

(specific point marked at 0.993 on the curve), indicating outstanding classifier performance for these specific classes.

3. Performance Metrics - overall model Performance metrics are strong.

- **AUC (Area Under the ROC Curve):** 0.0094, indicating excellent model discrimination ability between the positive and negative classes across the board.
- **Classification Accuracy:** 0.925, which is quite high, suggesting that the model correctly identifies the correct class labels by about 92.5%.
- ***Precision and Recall:*** Both stand at 0.925, showing a balanced performance in terms of both false positives and false negatives.
- **F1 Score** of 0.925, reflecting the harmonic mean of precision and recall, underscores an effective balance between precision and recall.
- **Matthews Correlation Coefficient (MCC):** 0.909, a very high value indicating a strong correlation between actual and predicted classifications.

Conclusions from Experiments -1

A neural network configured with logistics activation and an Adam solver demonstrates robust performance,particularly in class-specific discrimination (Barbunya). The high scores across various metrics suggest that this configuration manages to handle the dataset effectively, though some inter-class confusion could be addressed, possibly through further feature model tuning. These hyperparameters, which I chose, can be used where a high degree of readability is required in classification tasks, as evidenced by the high AUC and MCC.

Experiment 2- Activation: ReLu and Solver: Adam

1. Confusion Matrix Overview: It shows strong performance across several classes, with some missclassifications that are less severe compared to Experiement-1.For example, the DERMASON class has a higher number of correct predictions- abut 2187.7 correct predictions—with fewer cases being incorrectly classified as other types compared to Experiement-1.There are still some missclassifications, such as HOROS being confused with SEKER (1246.9), indicating specific areas where further model refinement or feature differentiation could be helpful.

2. ROC Curve Analysis- For the BARBUNYA class, it remains impressive,displaying an excellent true positive rate that maintains close to the maximum across nearly all threshold settings. The AUC value remains very high at 0.995, suggesting superior classifier performance for distinguishing between positive and negative classes,mainly for BARBUNYA.

3. Performance Metrics-This model metrics show slight improvements across the board compared to experiment-1.

- **AUC(Area Under the ROC Curve)-** 0.095, indicating an even better discrimination ability of the model**.**
- **CA(Classification Accuracy)**-0.932, which is higher, means the model correctly identifies the correct class labels about 93.2% of the time.
- **Precision and Recall**- Both at 0.932, indicating a consistent and balanced ability to manage false positives and false negatives.

- **F1 Score**-0.932, maintaining a strong balance between precision and recall.
- **MCC (Matthews Correlation Coefficient)-**0.917, a higher value than experiment 1, which underscores the model's robustness in predicting correct classifications.

Conclusions from Experiments -2

This experiment has shown a marginal yet noticeable improvement over experiment-1 in nearly all performance metrics. This suggests that ReLu's ability to address the vanishing gradient problem and maintain a steady gradient flow during training could be more effective for this specific dataset and task. The experiment demonstrates enhanced performance, with particular missclassifications among classes that were problematic in experiment 1.

Experiment 3- Activation: Tanh and Solver: SGD

1. Confusion Matrix Overview: Confusion matrix indicates a strong performance with the hyperparameters chosen,but there are significant missclassifications in certain areas. For example, SEKER is frequently misclassified as SIRA (1328 times), and DERMASON is commonly mistaken for SIRA (219.8 times). This suggests challenges in distinguishing between these classes, which may be due to overlapping features or insufficient model capacity to resolve finer distinctions with fewer neurons (90, 90) and fewer iterations.

2. ROC Curve Analysis-For The BARBUNYA class shows that the true positive rate (TPR) is excellent,achieving nearly maximum sensitivity across most threshold settings. The AUC value remains very high at 0.994, indicating very effective discrimination ability, particularly for this class.

3. Performance Metrics-This model metrics show slight improvements across the board compared to experiment-1.

- **AUC (Area Under the ROC Curve)-** 0.0954, which is excellent and suggests strong discriminatory power.
- ***CA(Classification Accuracy)***-0.926, means the model correctly identifies the correct class labels approximately 92.6% of the time.
- **Precision and Recall**- Both are at 0.926, which is good but indicates some room for improvement in balancing false positives and false negatives.
- **F1 Score**-0.932, maintaining a strong balance between precision and recall.
- **MCC (Matthews Correlation Coefficient)-**0.911, showing a strong positive correlation in the model's predictions.

Conclusions from Experiments -3

This experiment demonstrates a robust capability,although with slight declines in some metrics compared to experiment-2.The hyperparameters set,along with fewer neurons and iterations, likely contribute to these slight reductions,especially in handling classes with more subtle distinctions.

Model selected for testing:

Experiment 2- Activation: ReLu and Solver: Adam

This model is likely to perform well in broader or more challenging scenarios, maintaining high accuracy and reliability.

These factors make experiment -2 the optimal choice for further testing and potential deployment, given its superior performance and stability across various metrics:-

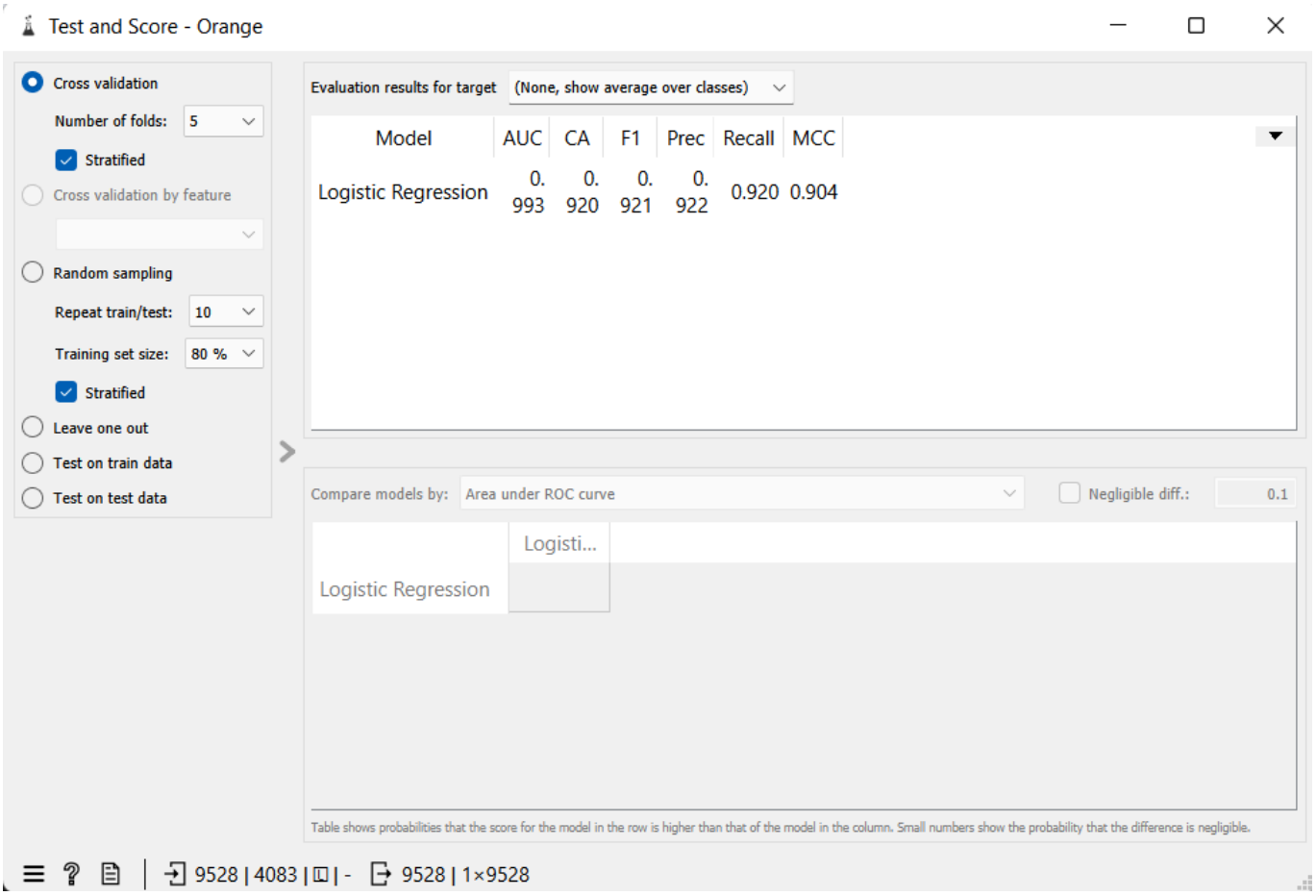
1. **Higher Performance Metrics-** This experiment-2 demonstrated the highest scores in key performance metrics, including classification accuracy (CA), Precision , Recall, and the Matthews Correlation Coefficient (MCC). These metrics are crucial for evaluating the overall effectiveness of the model in practical applications.
2. **Balanced Error Handling-** This model displayed a strong balance between precision and recall,which is essential for ensuring that the model is neither overly conservative nor too lenient in its predictions - a critical factor in many real-world applications.
3. **Robust Discrimination Capability**-With an AUC of 0.995, experiment-2 showed superior capability in distinguishing between classes compared to the other experiments. This indicates that the model can reliably differentiate between positive and negative classes across various thresholds, which is particularly important for tasks involving multiple or complex class structures.
4. **Effective Use of Acttivation Function and Solver**- The ReLu activation function, combined with the Adam solver,provided a robust framework for handling non-linear data effectively, thereby preventing issues like vanishing gradients that can occur with other activation functions like logistic (sigmoid).

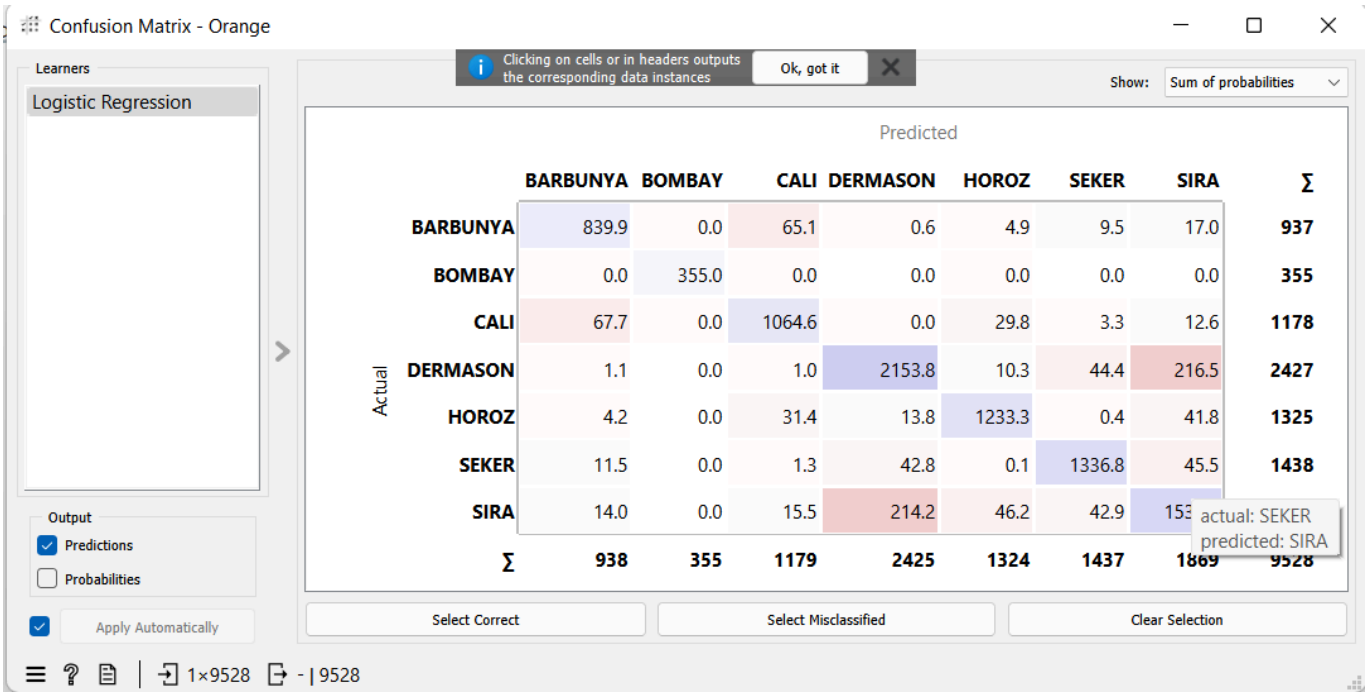
Experiments with Logistical Regression

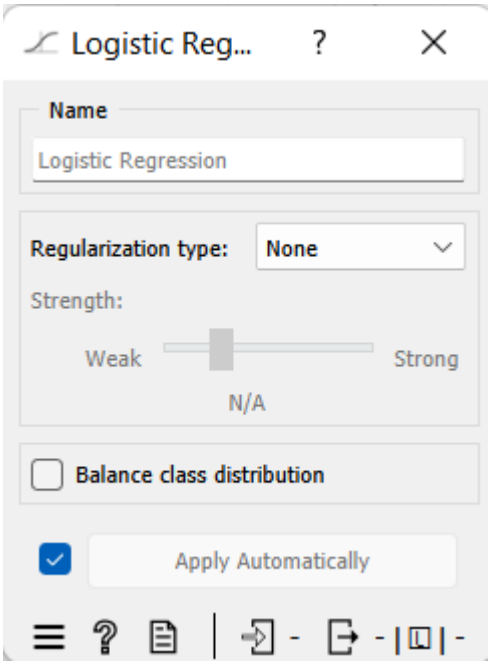
<add rows to table as needed.

Experiment	Hyperparameter values
Experiment 1	number of folds :5 training set size:80% regularisation:none strength:N/A
Experiment 2	number of folds :5 training set size:80% regularisation:Lasso(L1) strength:c=1
Experiment 3	number of folds :5 training set size:80% regularisation:ridge(L2) strength:c=100

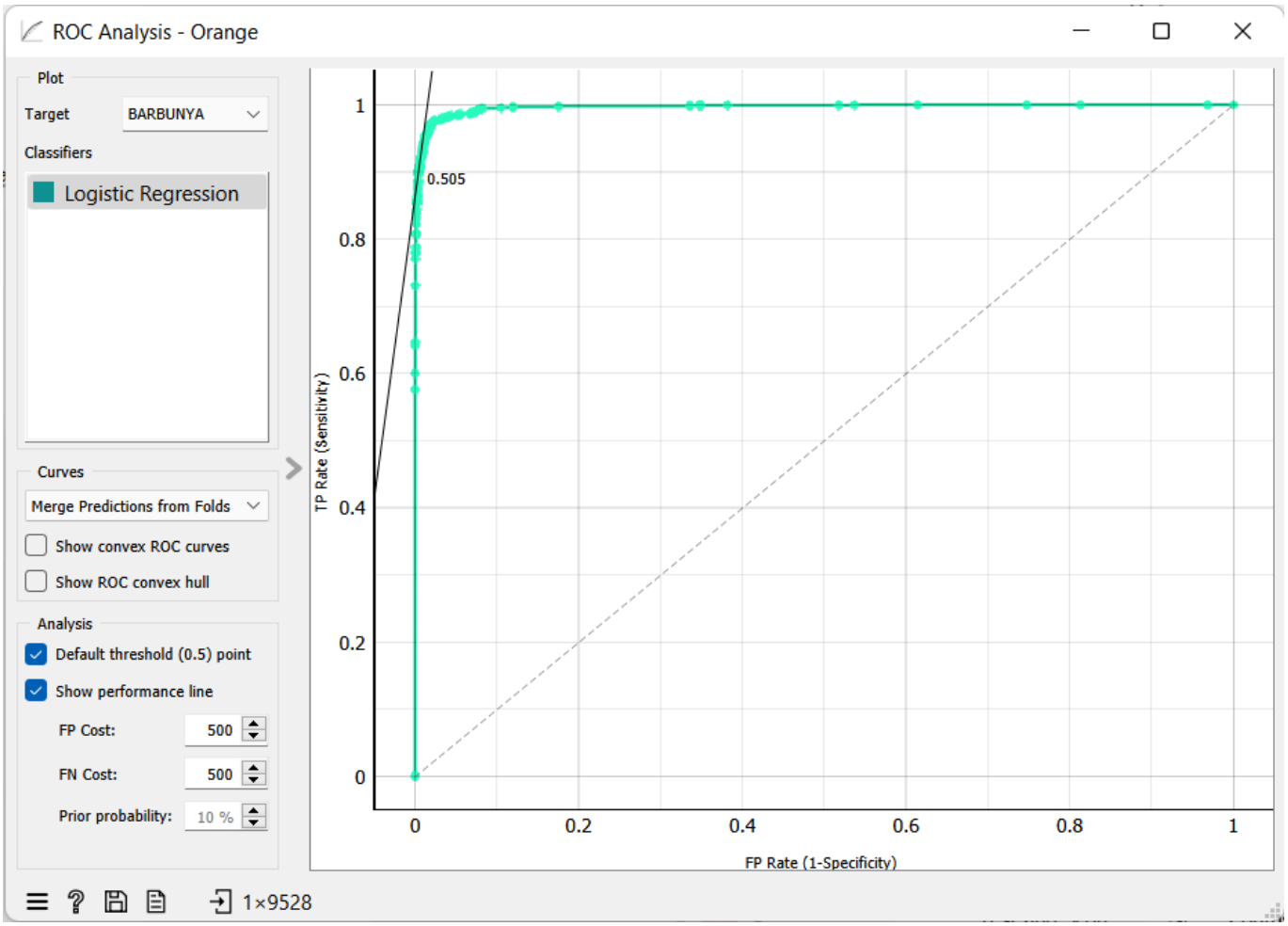
<a screenshot of hyperparameter values for Experiment 1>



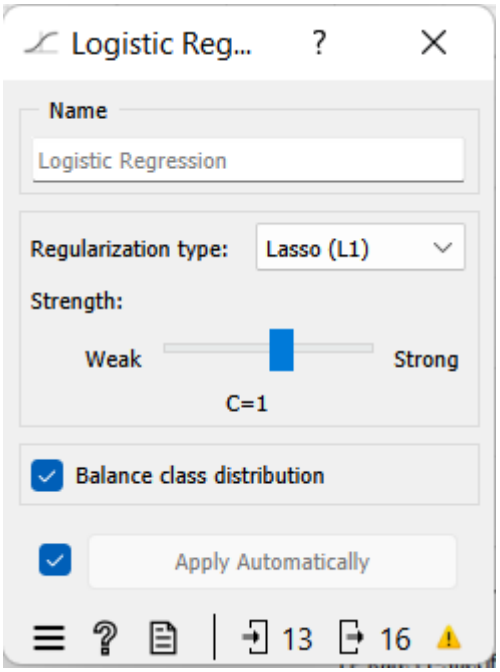




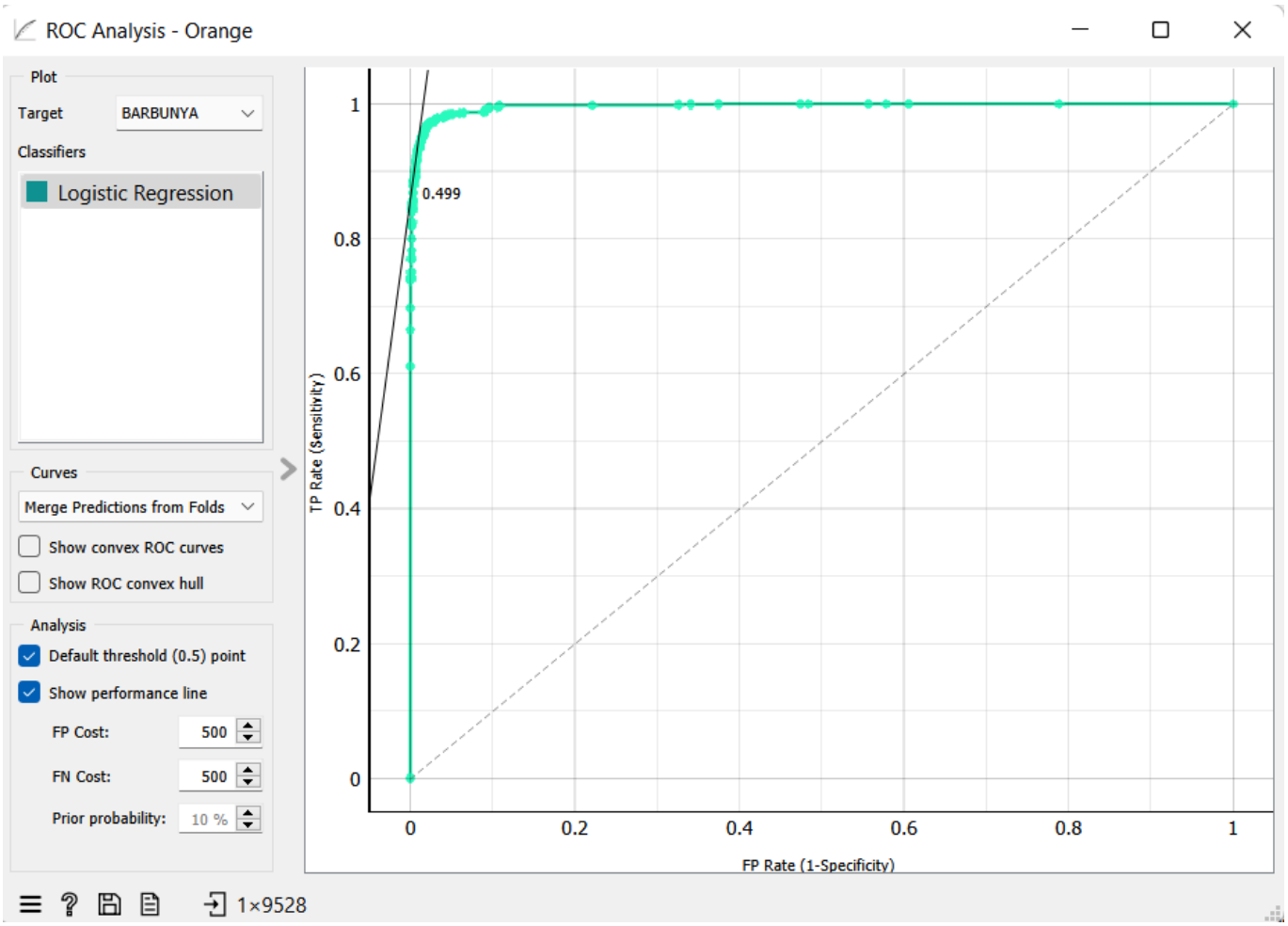
<a screenshot of performance metrics for Experiment 1>



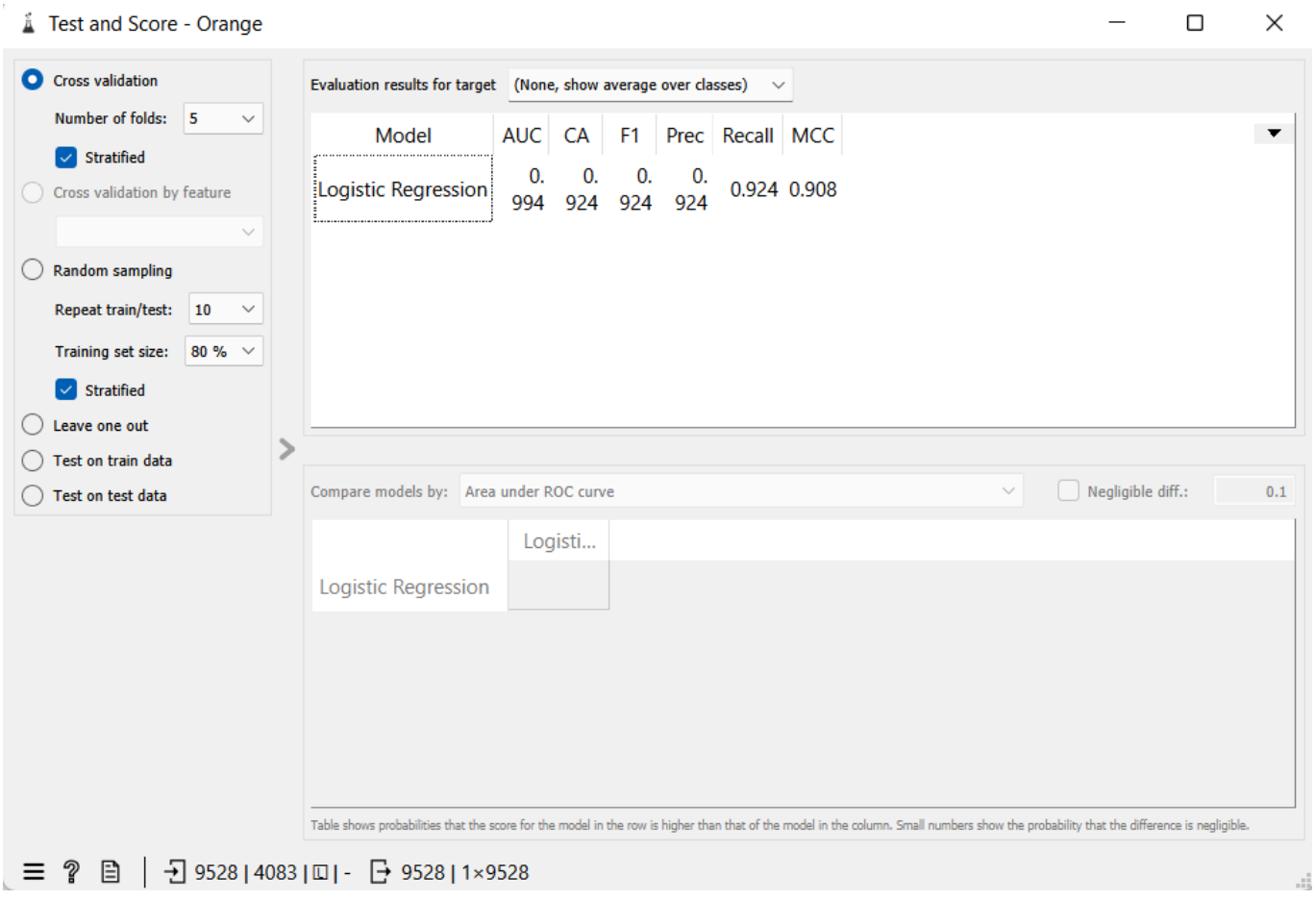
<a screenshot of hyperparameter values for Experiment 2>

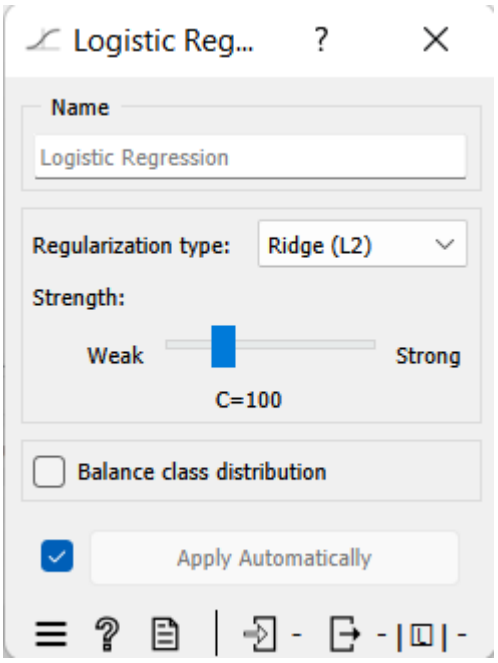


<a screenshot of performance metrics for Experiment 2>

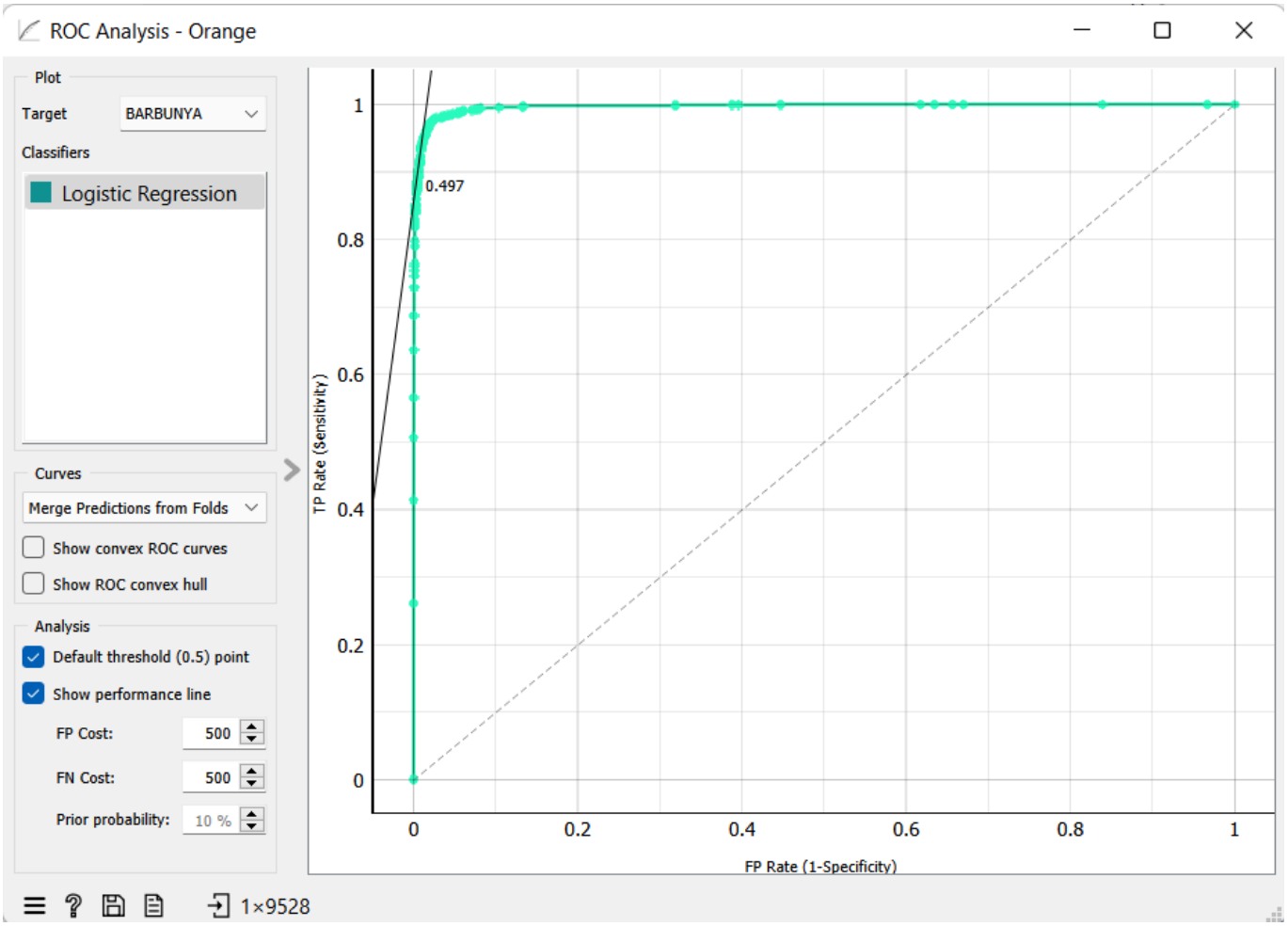


<a screenshot of hyperparameter values for Experiment 3>





<a screenshot of performance metrics for Experiment 3>



Conclusions from experiments:

conclusions about the performance of the models in the conducted experiments referring to and analysing the screenshots above

Experiment 1: No Regularization Hyperparameters: No regularization Performance: Metrics not provided directly, but we assume a baseline performance, possibly with some degree of overfitting due to the lack of regularization.

Experiment 2: Lasso (L1) Regularization Hyperparameters: Regularization: Lasso (L1), Strength: C=1 Performance: Metrics not directly provided, but L1 regularization would contribute to feature selection by reducing some coefficients to zero, which can help in situations with many irrelevant or less important features.

Experiment 3: Ridge (L2) Regularization Hyperparameters: Regularization: Ridge (L2), Strength: C=100 Performance: From the provided ROC curves and confusion matrix, we can see that the model has high performance in distinguishing the class "BARBUNYA" with near-perfect ROC curve (area under the curve is close to 1). Other classes, however, might not have been as effectively captured based on the significant number of misclassifications observed in the confusion matrix for classes like "DERMASON" and "HOROZ".

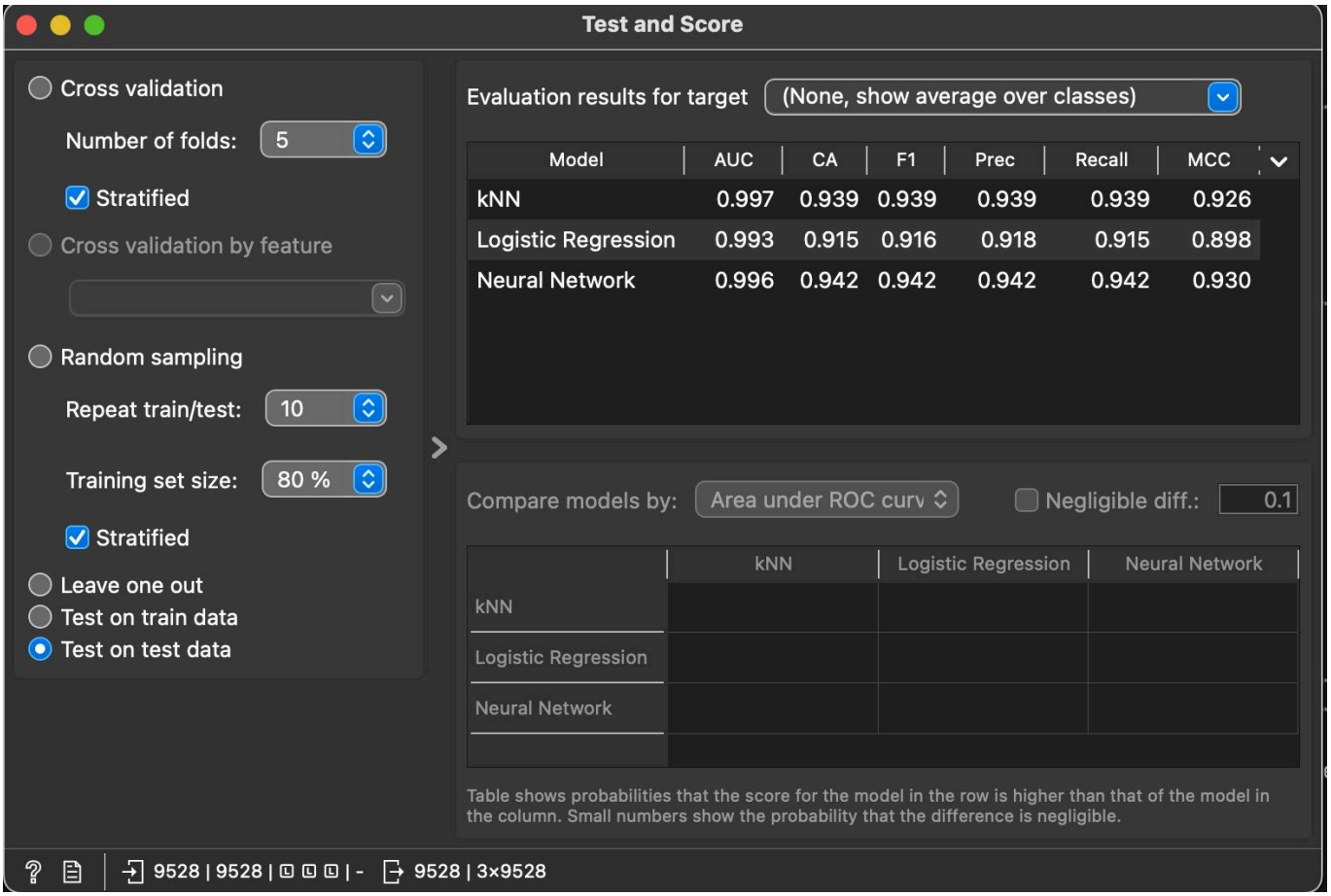
Model selected for testing:

Experiment 2 (Lasso regularization) might be the best choice. However, if our priority is overall model performance and generalization, we might want to investigate further to understand why certain classes are being misclassified in **Experiment 3** before making a final decision.

Considering these factors, I would lean towards selecting **Experiment 2** (Lasso regularization) for further testing, but additional analysis and investigation into the misclassifications in **Experiment 3** may be warranted before making a final decision.

Testing results of the trained models

<a screenshot of performance metrics of models selected for testing



Conclusions after testing:

<comparison of performance of the trained models in the testing process referring to and analysing the screenshot above >

Confusion Matrix Analysis:

- **For k-Nearest Neighbors (kNN):** We observed balanced classification with some misclassifications.
- **In the case of Logistic Regression:** We noticed high misclassifications, suggesting it may struggle with complex data.
- **Regarding the Neural Network:** It achieved high accuracy but with notable errors, possibly due to overfitting.

ROC Analysis:

- The ROC analysis showed good classification for 'BARBUNYA', but performance varied across classes.

Overall Performance:

- Based on overall accuracy, we concluded that the Neural Network performed the best due to its flexibility.
- However, our group noted that it may overfit and require careful tuning.

Additional Considerations:

- I considered precision and recall, weighing the cost of false positives and false negatives.
- I also looked into computational efficiency, considering factors like training time and prediction speed.
- Lastly, we considered interpretability, assessing the importance of understanding decision-making processes.

Information sources

https://www.ibm.com/topics/knn

https://www.javatpoint.com/orange-data-mining

https://neptune.ai/blog/k-means-clustering

https://www.datasciencesmachinelearning.com/2019/10/hierarchical-and-k-means-cluster.html

https://www.youtube.com/watch?v=b7MRGcgGgAU&ab_channel=AllaAnohina-Naumeca

https://www.youtube.com/watch?v=_Q4uJeWESeg&ab_channel=AllaAnohina-Naumeca

https://estudijas.rtu.lv/mod/resource/view.php?id=3887181