**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: **14**

Students:

Danylo Melnyk (221ADB195)

Tuna Yalcin (221ADB180)

Omar Ahmed Abdelshafi Ahmed Mohamed (240AEB049)

Lucas Le Porhiel(240AEB029)

Lucas Leblanc(240AEB028)

Rana Irem Turhan(221ADB049)

**Teaching staff:** Alla Anohina-Naumeca

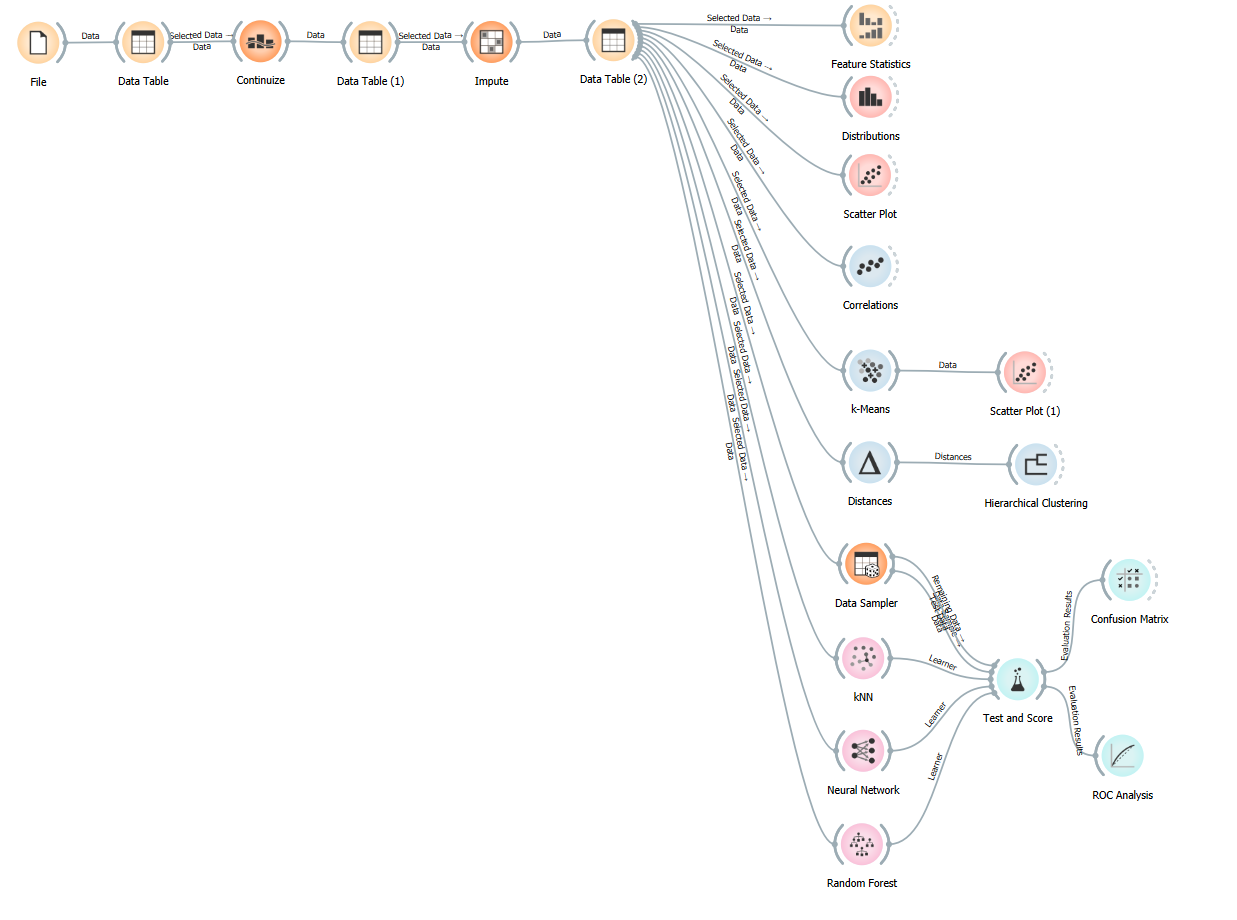
**Project link:** <https://github.com/Danylo-Melnyk-Git/Second-practical-assignment.git>

**Link to dataset:**

<https://www.kaggle.com/datasets/tawfikelmetwally/automobile-dataset>

2023/2024 academic year

# Orange tool workflow



# Part I

## Description of the dataset

### **Dataset Title:** Car information dataset

### **Dataset source:** [Car information dataset (kaggle.com)](https://www.kaggle.com/datasets/tawfikelmetwally/automobile-dataset)

### **Creator and/or owner of the dataset:**

**Author:** Ross Quinlan

**Published by:** UCI Machine Learning Repository

**DOI:** <https://doi.org/10.24432/C5859H>

***Description of the dataset problem domain:***

The dataset contains comprehensive information about automobiles, covering various aspects crucial to understanding their performance, characteristics, and origins. It includes data on fuel efficiency measured in miles per gallon (MPG), the number of cylinders in the engine, engine displacement indicating size or capacity, horsepower, weight, acceleration capability, model year of manufacture, and the country or region of origin. Analyzing and exploring this dataset enables a comprehensive exploration of the automotive industry, informing stakeholders ranging from manufacturers and designers to policymakers and consumers.

***Dataset licensing conditions:***

This dataset is licensed under a [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/legalcode) (CC BY 4.0) license.

This allows for the sharing and adaptation of the datasets for any purpose, provided that the appropriate credit is given.

***Information about the method or procedure for collecting the dataset,***

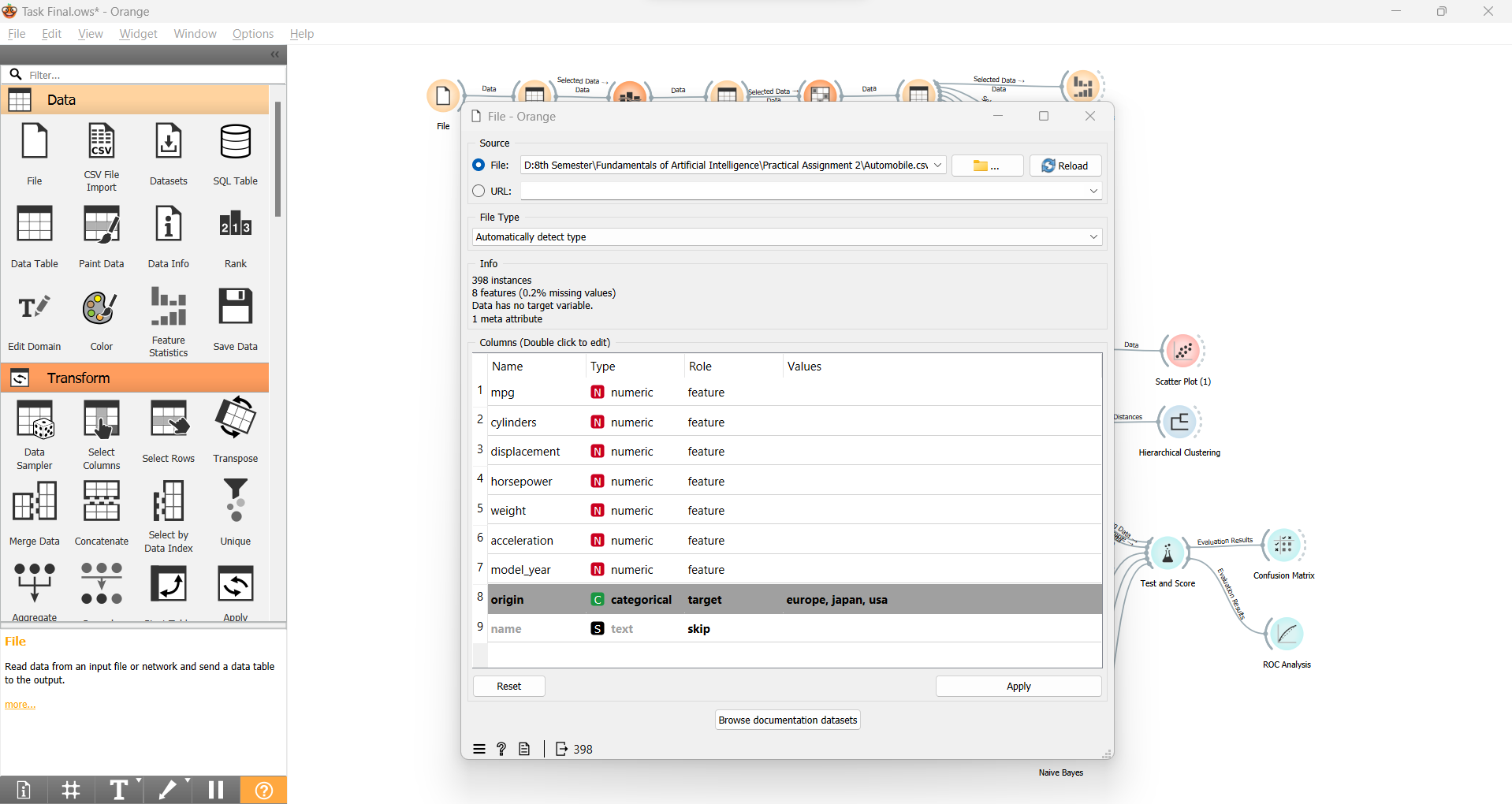
***Collection Methodology:***

This original dataset was taken from the StatLib library, which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American Statistical Association Exposition.

## Description of the dataset content

*Number of data objects in the dataset:* **398**

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



### Number of classes in the dataset: **3**

*Description of classes:*

It represents the country or region of origin for each automobile.

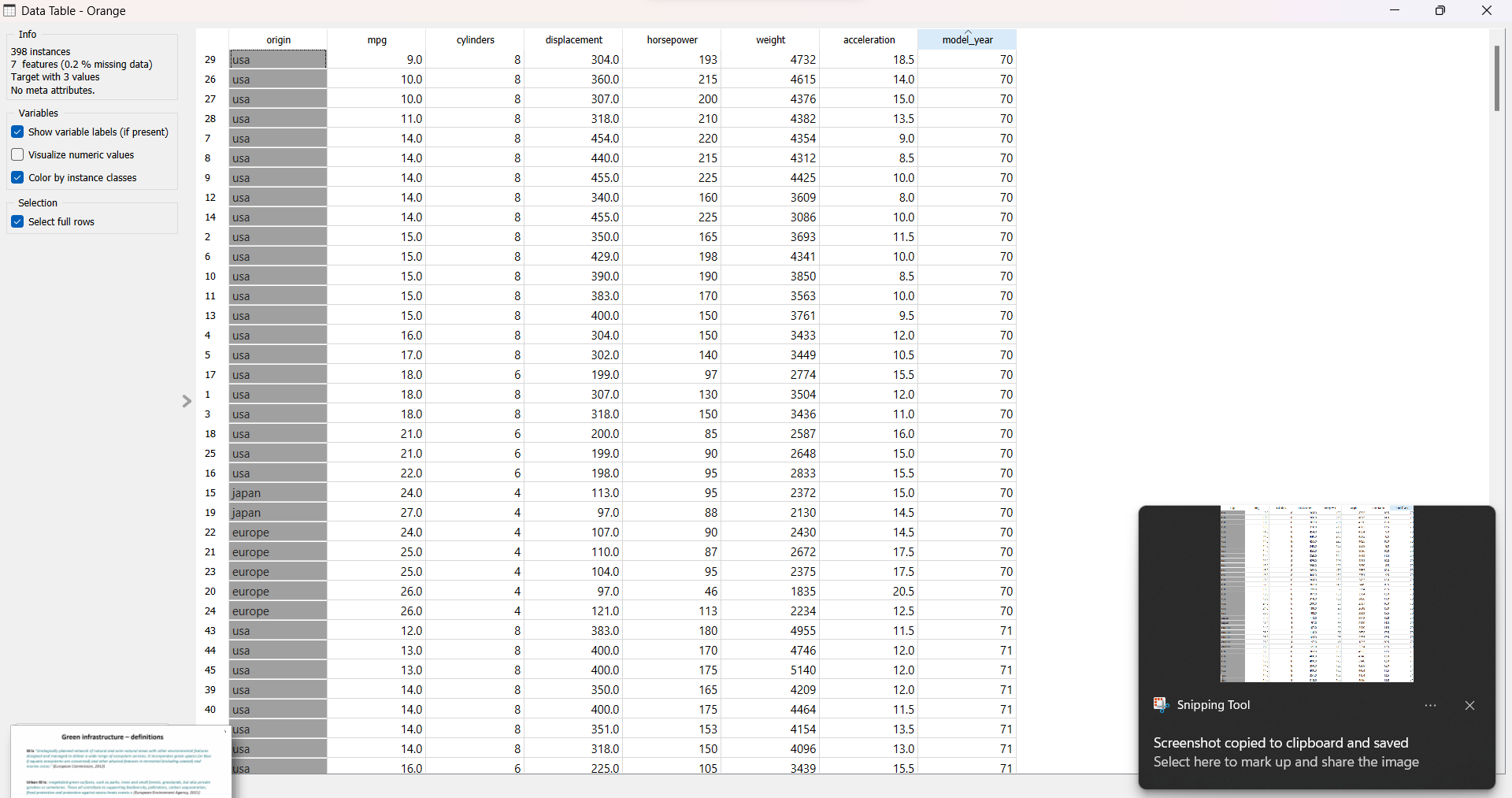
*Number of data objects belonging to each class:*

| **Class label** | **Number of data objects** |
| --- | --- |
| usa | 249 |
| japan | 79 |
| europe | 70 |

### Description of features:

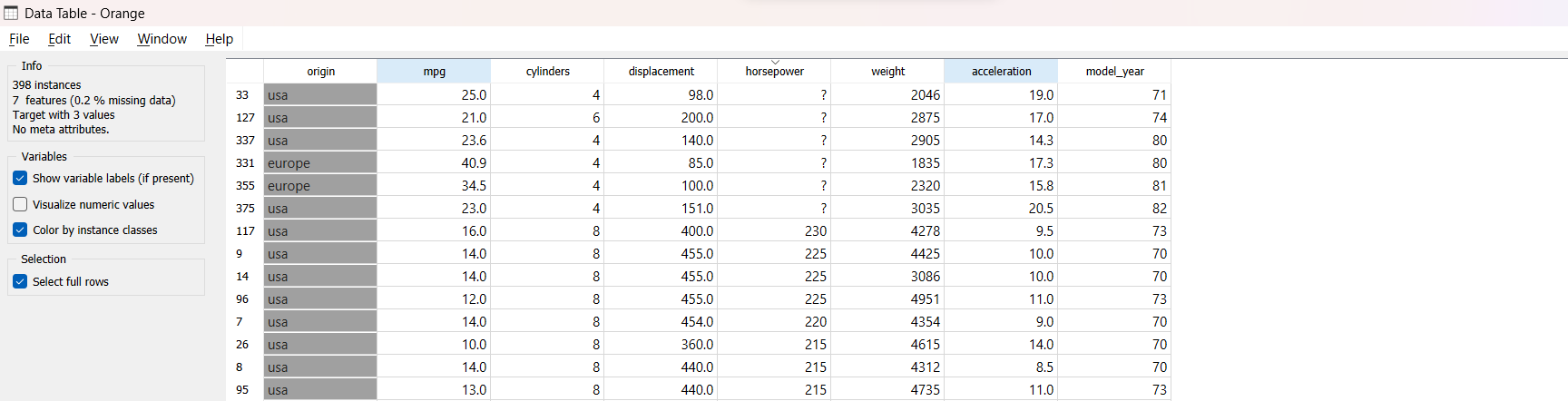
| **Feature title** | **Explanation of the feature** | **Value type** | **Range of values** |
| --- | --- | --- | --- |
| mpg | Miles Per Gallon (fuel efficiency) | Continuous | 9 - 46.6 |
| cylinders | Number of cylinders in the engine | Integer | 3 - 8 |
| displacement | Engine displacement in cubic inches. | Continuous | 68 - 455 |
| horsepower | Horsepower of the engine | Continuous | 46 - 230 |
| weight | Weight of the car in pounds | Continuous | 9 - 46.6 |
| acceleration | Acceleration rate (0-60 mph time) in seconds | Continuous | 1613 - 5140 |
| model\_year | Year of manufacture for the automobile model (e.g., 70 for 1970) | Integer | 70 - 82 |

### Data file structure:



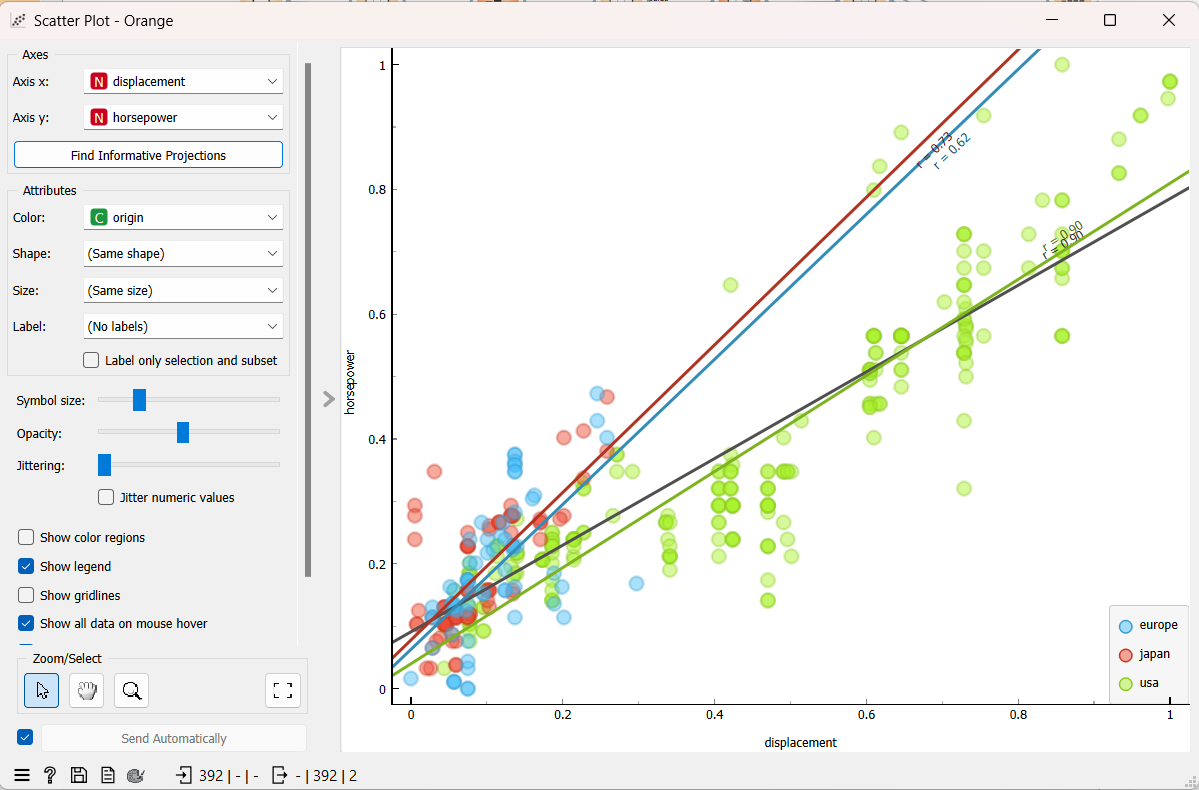
### Information about missing values or outliers:

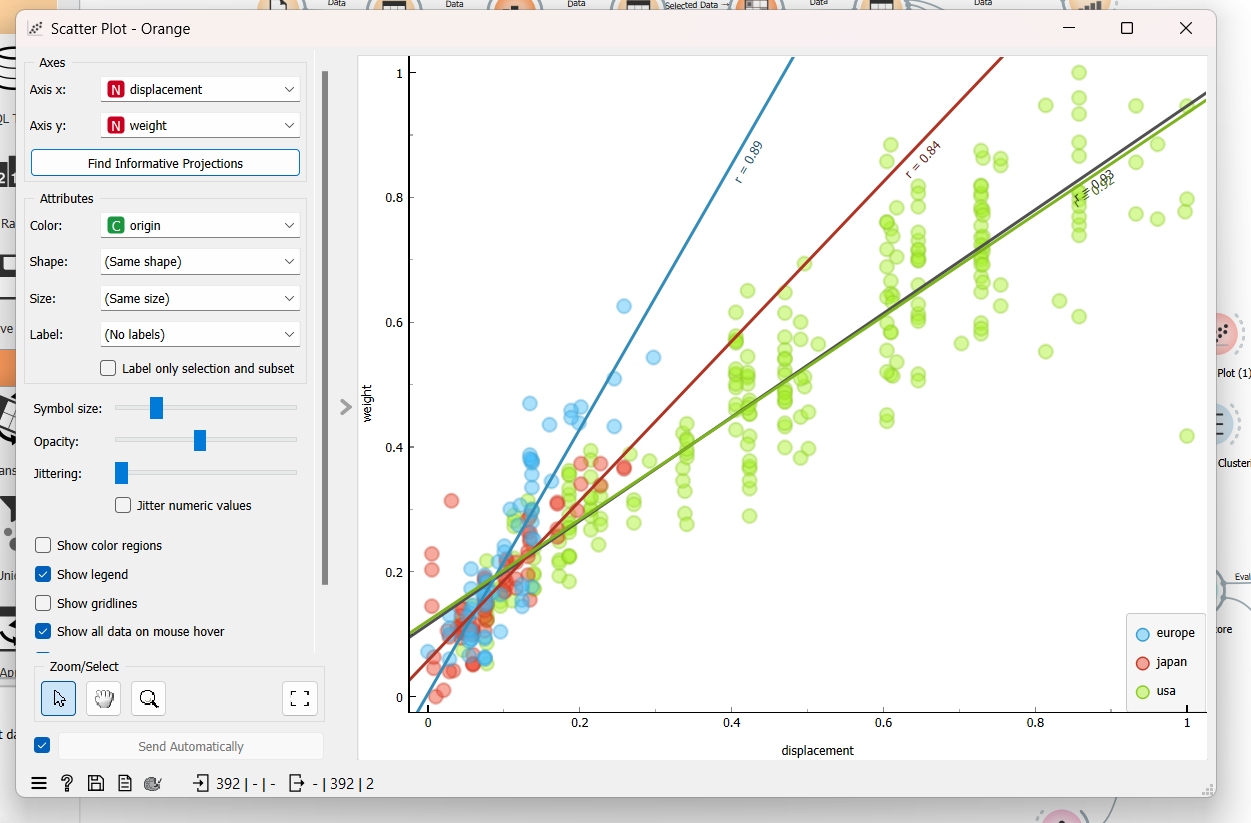
Horsepower has 6 missing values which represent around 0.2% of the total number of objects as shown below.



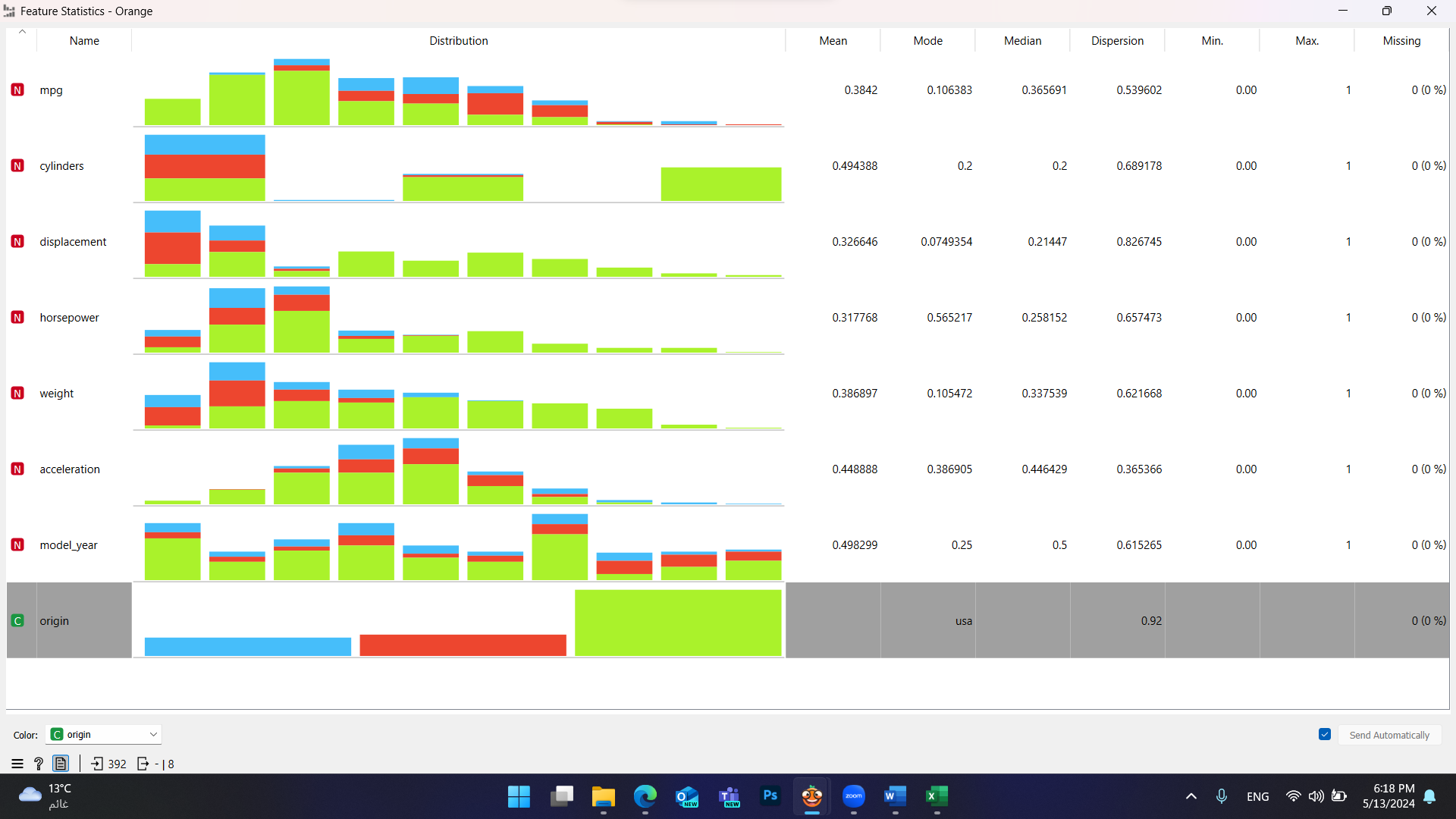
## Visual and statistical representation of the dataset

<a scatterplot screenshot>

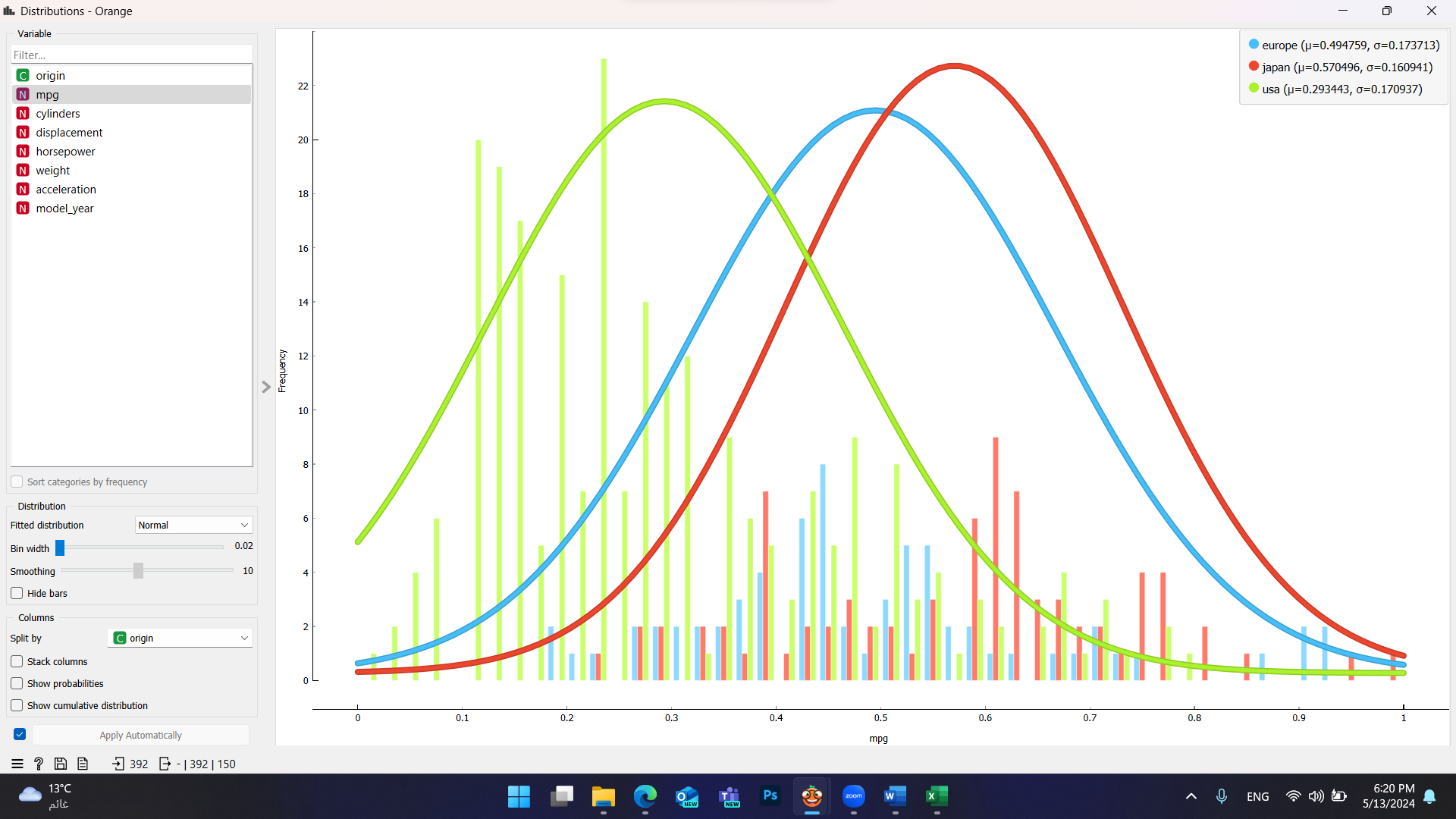


<a scatterplot screenshot>

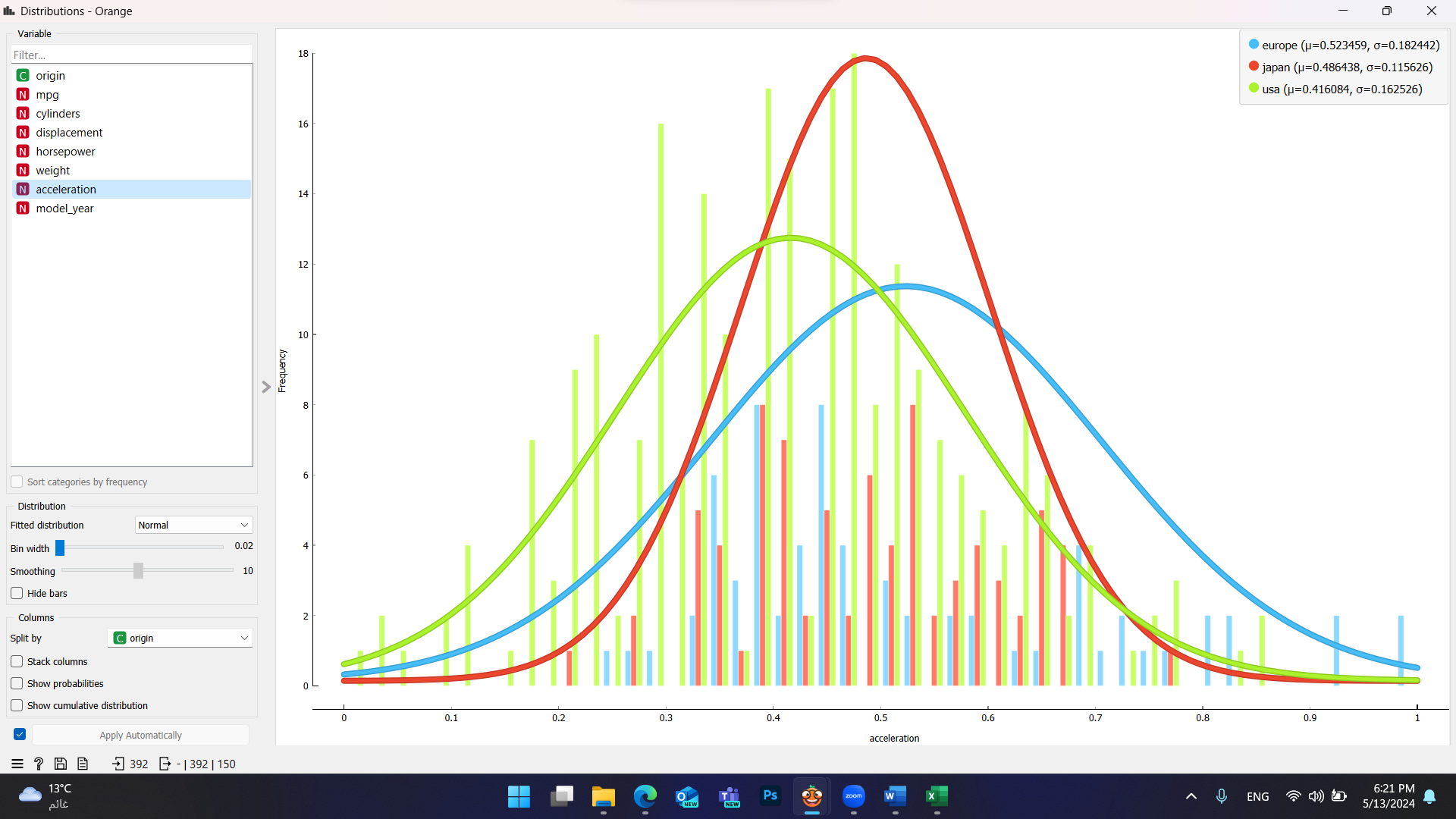
<a histogram screenshot>



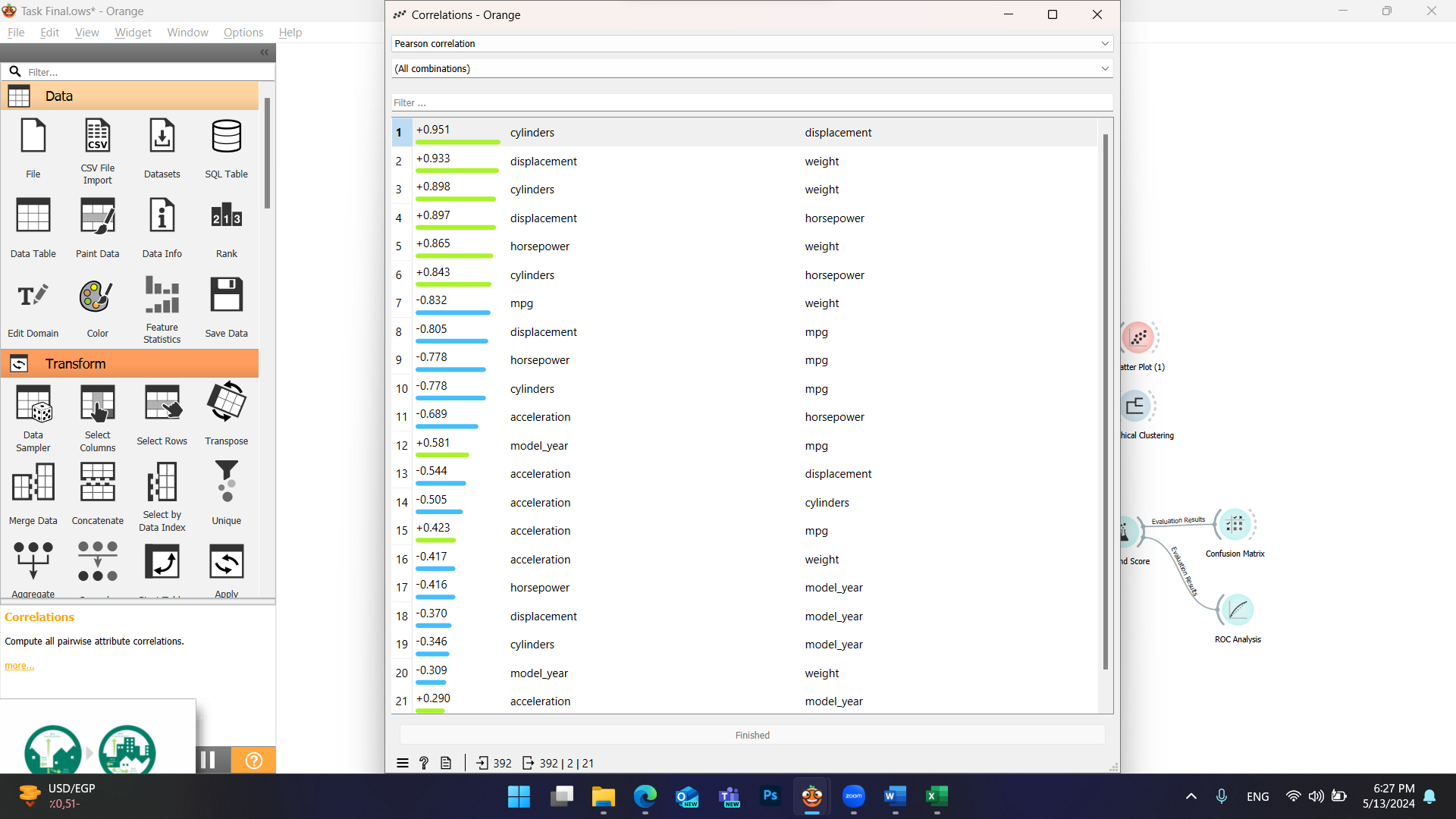
<a screenshot of feature distribution>



<a screenshot of feature distribution>



<a screenshot with statistics>



## Answers to questions

<answers the questions below, referring to the screenshots above and providing an analysis of the results>

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

No, they are not balanced because one country which is the USA has significantly more samples compared to others.

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

Yes, visual representation of data can indeed provide valuable insights into the structure of the dataset.

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

The visual representation of the data shows us there are two groups.

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

No, the clusters overlap and are close to each other.

## Conclusions arising from the analysis of statistical indicators

<analysis of statistical indicators by referencing specific values>

The analysis of statistical indicators reveals significant insights into the relationships within the automobile dataset. Strong positive correlations are evident between certain pairs of variables, notably between cylinders and displacement (r = +0.951), displacement and weight (r = +0.933), and cylinders and weight (r = +0.898). These high positive correlations signify a direct relationship, suggesting that as one variable increases, the other tends to increase as well, indicating a strong linear association. Conversely, strong negative correlations are observed between mpg and weight (r = -0.832), displacement and mpg (r = -0.805), and horsepower and mpg (r = -0.778), indicating an inverse relationship. For instance, as vehicle weight increases, fuel efficiency, represented by mpg, tends to decrease.

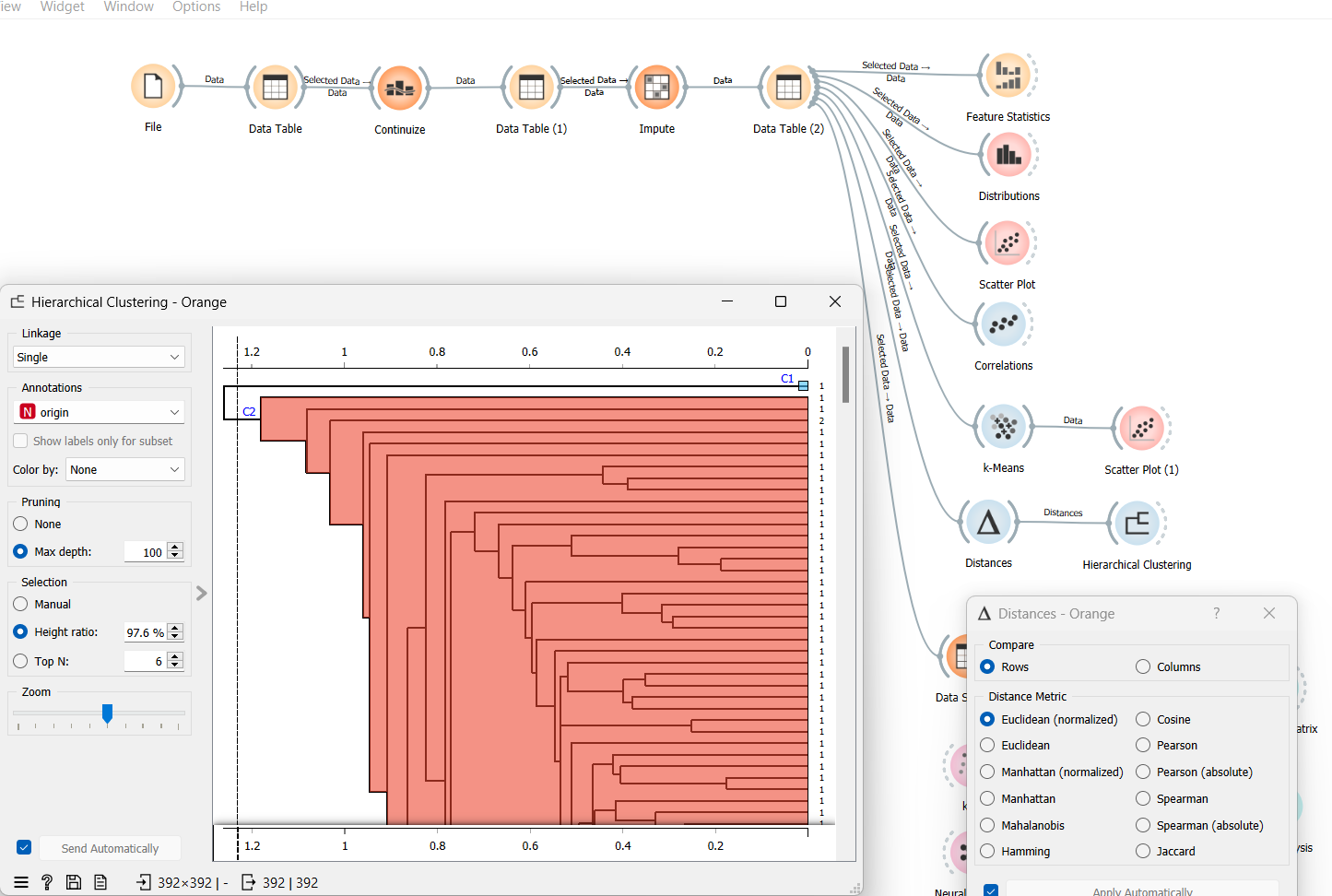
# Part II

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

## Hierarchical clustering

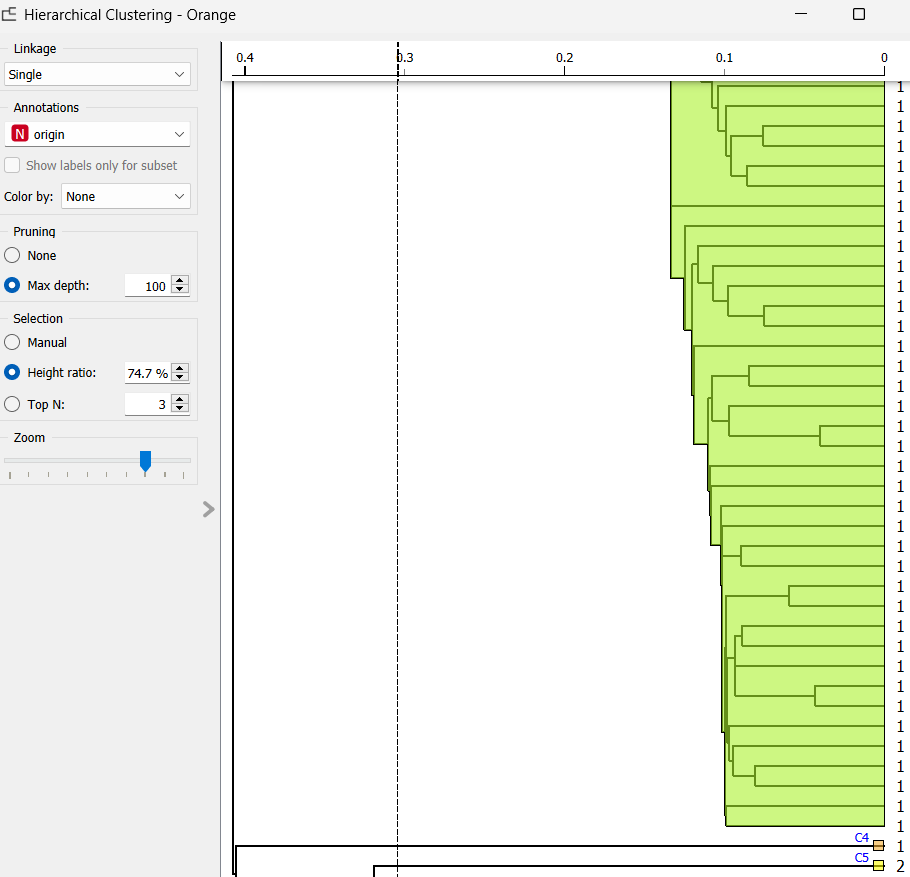
### Hyperparameters available in the Orange tool:

| **Hyperparameter** | **Description** |
| --- | --- |
| Linkage method | The distance between clusters is calculated using the linkage criterion. There are some options like “Single”, “Average linkage: computes the average distance between data objects of two clusters”, “Complete: computes the distance between the clusters’ most distant data objects”..etc. We have used Single Linkage (nearest neighbor) in order to compute the distance between the closest data objects of two clusters.  REF:[javatpoint](https://www.javatpoint.com/hierarchical-clustering-in-machine-learning#:~:text=The%20agglomerative%20hierarchical%20clustering%20algorithm%20is%20a%20popular%20example%20of,closest%20pair%20of%20clusters%20together) |
| Distance Metric | The distance metric is used to determine the distance between instances. It has several options like “Euclidean”, “Manhattan”, “Hamming”.. Etc. We have used the Euclidean distance, which is a “straight line” representing the shortest distance between two points. |
| Pruning | By selecting the maximum depth, huge dendrograms can be pruned, but this is only for display purposes, not the actual clustering.  REF::[orange](https://orangedatamining.com/widget-catalog/unsupervised/hierarchicalclustering/) |
| Annotation | Labels of nodes in the dendrogram can be chosen in the annotation box  REF:[orange](https://orangedatamining.com/widget-catalog/unsupervised/hierarchicalclustering/) |
| Selection | 1. Manual selection can be done by clicking inside the dendrogram, we can select the clusters, and each cluster is shown by a different color.  2. Height ratio: The line that we vertically move inside the dendrogram selects the items of the right side which separates the data objects into particular classes, which determines how the algorithm decides when to stop merging clusters and reflects the number of clusters we intend to create.  3. Top N selects the number of top nodes.  REF:[orange](https://orangedatamining.com/widget-catalog/unsupervised/hierarchicalclustering/) / [ORTUS](https://estudijas.rtu.lv/course/view.php?id=361245&section=10#h5pbookid=21751&section=top&chapter=h5p-interactive-book-chapter-cf0bea7e-d0ff-4196-a030-59fe067a7963) |

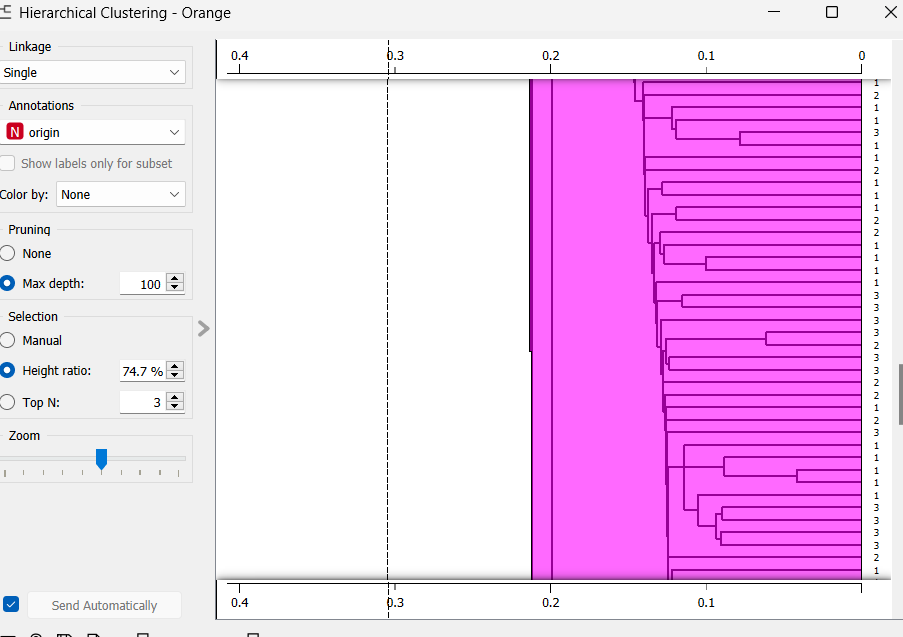


### Description of experiments

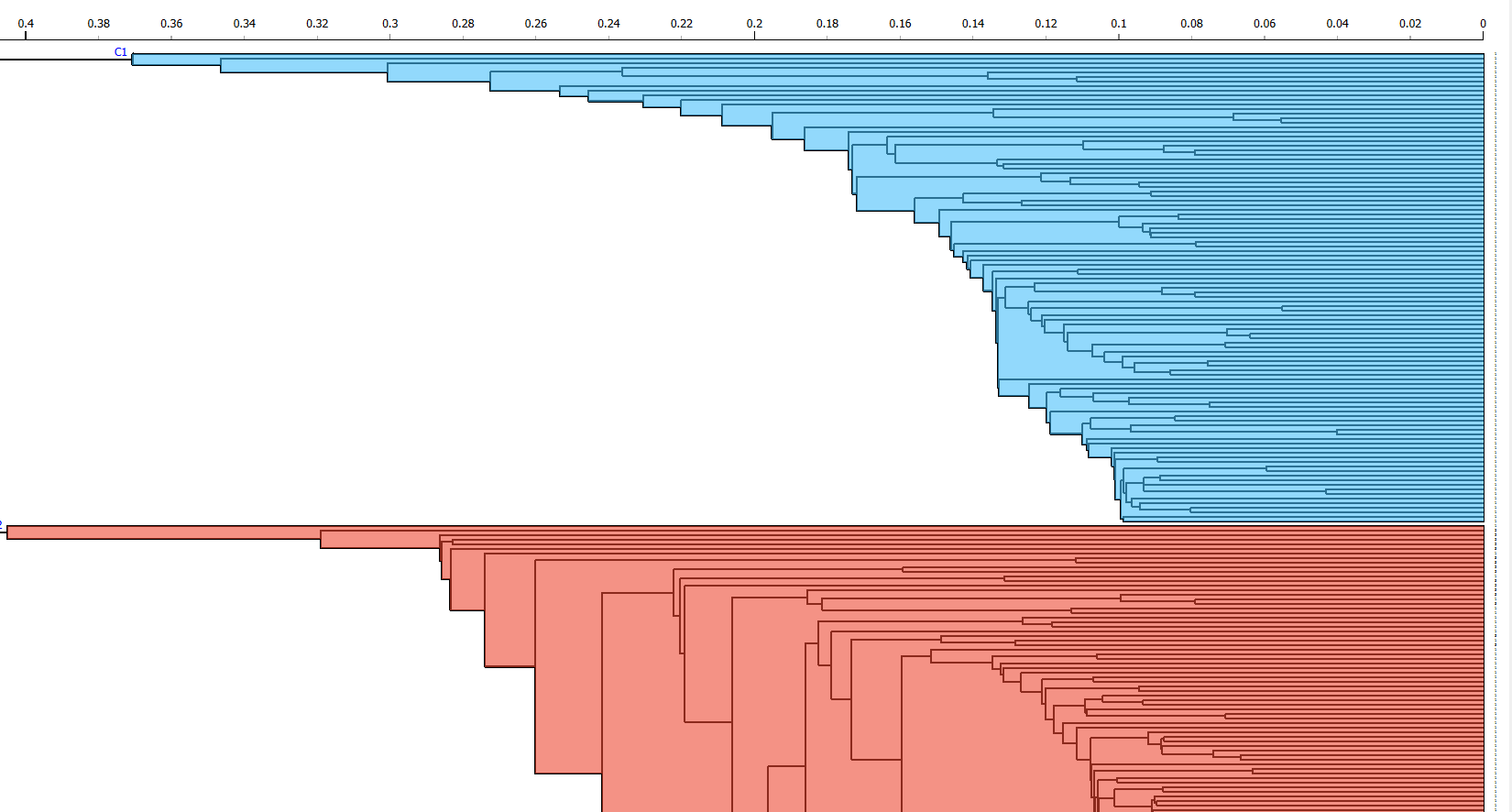
1.



Here we can see one big cluster C3 containing several same data objects together but below we can see there are two clusters C4 and C5 containing only one data object inside.

2.

Here we can see a cluster C6 with a mix of cars that have different origins (it is not a good separation).

3.

When we moved the line to the left side, we noticed two big clusters: the blue one contained data objects which had the same origin (target attribute), while the red one consisted of data objects which had different origins.

### Conclusions from experiments:

This superficial experiment showed that some data objects remained isolated and others formed distinct clusters. Overall, the experiment highlights how crucial it is to choose a suitable cut-off threshold when using hierarchical clustering. The exploration of various cluster structures and the identification of patterns that underlie in the data can be organized by modifying the cut-off line.

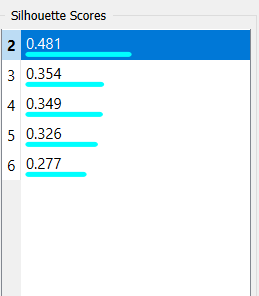
## K-means algorithm

### Hyperparameters available in the Orange tool:

| **Hyperparameter** | **Description** |
| --- | --- |
| Number of Clusters | It defines the number of distinct clusters that the algorithm will form. There are two options Fixed(clusters data to a specified number of clusters) and From X to Y (widget shoes clustering scores for the selected cluster range using the silhouette score)  Ref:[orange](https://orangedatamining.com/widget-catalog/unsupervised/hierarchicalclustering/) |
| Preprocessing | If the option is selected, columns are normalized.  Ref:[orange](https://orangedatamining.com/widget-catalog/unsupervised/hierarchicalclustering/) |
| Initialization | Choose an initialization method that favors convergence towards the optimal centroids of the clusters by strategically positioning the initial centroids before launching the iterative algorithm. |
| Re-runs/ Maximum iterations | Re-runs: how many times the algorithm is run from random initial positions.  Maximum iterations: the maximum number of iterations within each algorithm run.  Ref: [orange](https://orangedatamining.com/widget-catalog/unsupervised/hierarchicalclustering/) |

## 

### Description of experiments



After adjusting the K means, we obtained the highest Silhouette score of 0.481 which indicates the better-defined clusters. Upon exploring the orange tool, we found out that the MPG and acceleration have the closest silhouette scores.



In this scatter plot, we have observed clusters forming based on the values of the acceleration and the MPG.

### Conclusions from experiments:

Upon analyzing the scatter plot and taking into account the acceleration and MPG attributes in particular, We can conclude that there might be some degree of separability among the data points. However, one point to remember is that the degree of separability can vary across the different parts of the plot.

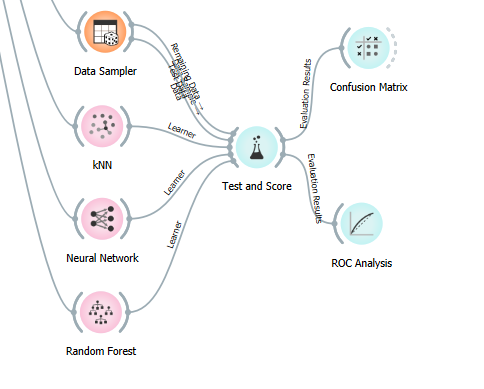
In conclusion, while the preliminary exploration using the K-means clustering and silhouette analysis indicates revealing underlying patterns within the automobile dataset, further validation is necessary to make definitive conclusions.

## Final conclusions

Based on the analysis of the experiments of hierarchical clustering and the K-means algorithm, the classes within the dataset may require further refinement for clear delineation. However, we were able to make the formation of distinct clusters through experimentation with different algorithms. Additionally, after working with the orange tool and conducting all the experiments, we can conclude that our chosen dataset is suitable for the practical parts and meets all the conditions required.

# Part III

Supervised machine learning algorithms, such as k-Nearest Neighbors (k-NN) and Naive Bayes, are valuable tools for analyzing the automobile dataset provided. In this context, the dataset includes features like miles per gallon (MPG), number of cylinders, displacement, horsepower, weight, acceleration, and model year. The target variable being the origin of the automobile, k-NN, known for its simplicity and effectiveness, could be applied to predict the origin of automobiles based on these features. By calculating the distances between data points in the feature space, k-NN identifies the k nearest neighbors and predicts the origin based on a majority vote among these neighbors. On the other hand, Naive Bayes, with its probabilistic framework, could also be utilized for classification.



## Description of the selected algorithms

Both k-NN and Naive Bayes present unique strengths depending on the task at hand. k-NN is valued for its simplicity, particularly beneficial for newcomers or situations where interpretability is paramount. Its non-parametric approach enables it to manage intricate data relationships adeptly, effectively capturing local patterns. Additionally, the lack of a conventional training phase in k-NN allows it to adapt to evolving data distributions over time.

In contrast, Naive Bayes offers simplicity and computational efficiency, making it particularly suitable for extensive datasets. Its probabilistic framework supports clear interpretation of outcomes and estimation of uncertainty. Furthermore, the assumption of feature independence in Naive Bayes proves advantageous for datasets with abundant irrelevant or redundant features. Particularly noteworthy is its proficiency in text classification tasks, such as spam detection and sentiment analysis.

### Title of the first algorithm: The k-Nearest Neighbors algorithm (k-NN)

### Description of the first algorithm:

The k-Nearest Neighbors (k-NN) algorithm is a simple and intuitive method used for both classification and regression tasks in machine learning. It's a type of instance-based learning algorithm, meaning it doesn't explicitly learn a model from the training data. Instead, it memorized the entire training dataset and used it during the prediction phase. For regression tasks, instead of assigning class labels, the k-NN algorithm predicts the target value for the new data point by averaging (or taking another aggregation) the target values of its k-nearest neighbors.

### Title of the second algorithm: Naive Bayes algorithm

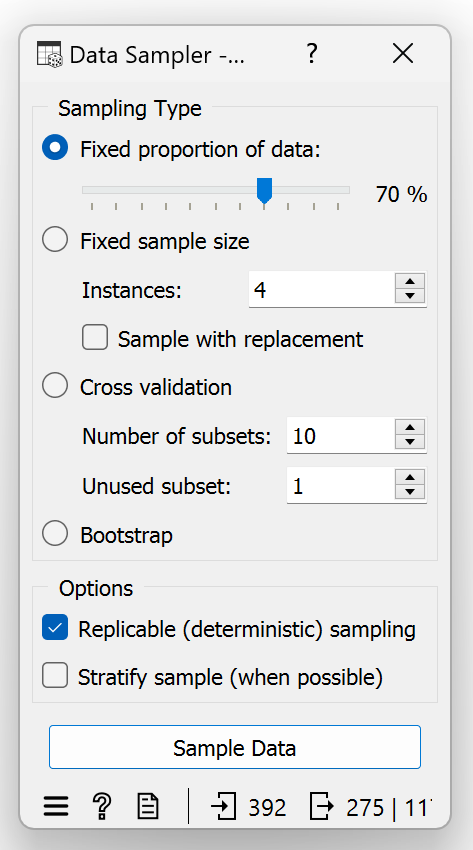
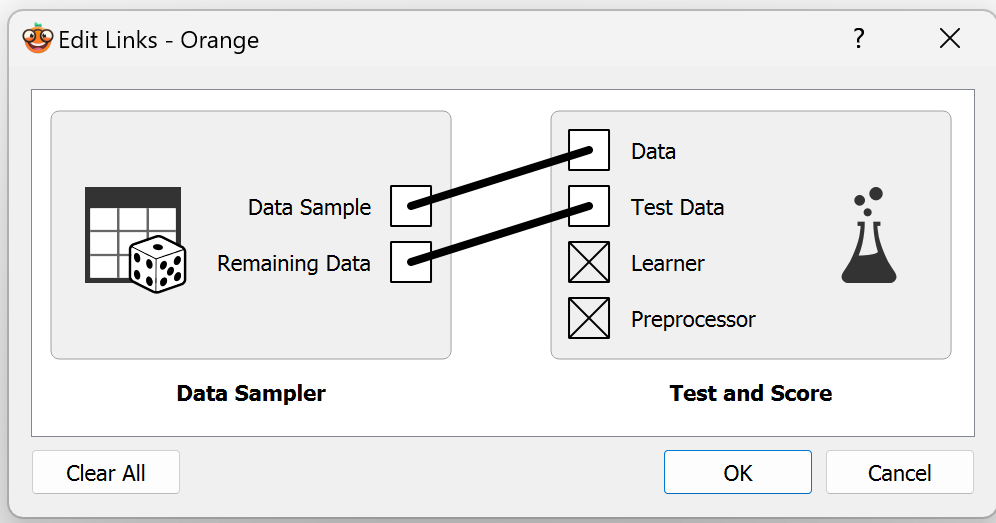
### Description of the second algorithm:

The Naive Bayes algorithm is a simple and probabilistic classification algorithm based on Bayes' theorem with the “naive” assumption of feature independence. Despite its simplicity, Naive Bayes can be surprisingly effective in many real-world classification tasks. In the context of an automobile dataset, this means that Naive Bayes assumes that the features (such as MPG, cylinders, displacement, horsepower, weight, acceleration, and model year) are independent of each other given the class label (in this case, the origin of the automobile).

## Description of hyperparameters

| **Hyperparameter** | **Description and values** |
| --- | --- |
| **Artificial Neural Networks** | |
| Neurons per hidden layer | defined as the ith element representing the number of neurons in the ith hidden layer. E.g., a neural network with 3 layers can be defined as 2, 3, 2. |
| Activation | Identity: no-op activation, useful to implement linear bottleneck  Logistic: the logistic sigmoid function  tanh: the hyperbolic tan function  ReLu: the rectified linear unit function |
| Solver | L-BFGS-B: an optimizer in the family of quasi-Newton methods  SGD: stochastic gradient descent  Adam: stochastic gradient-based optimizer |
| Regularisation, Alpha | L2 penalty (regularisation term) parameter |
| Max iterations | maximum number of iterations |
| **The k-Nearest Neighbours** | |
| Number of neighbors (n\_neighbors) | Number of neighbors to consider when making predictions. Higher values may result in smoother decision boundaries, while lower values may lead to more flexible boundaries. |
| Distance metric (metric) | Metric used to measure the distance between data points, such as Euclidean distance or Manhattan distance. The choice of metric can affect the performance of the algorithm. |
| Weights (weights) | Determines the weight of each neighbor when making predictions. |
| **Random Forest** | |
| Basic properties | Number of trees: Specify how many decision trees will be included in the forest.  Number of trees considered at each split: Specify how many attributes will be arbitrarily drawn for consideration at each node. If the latter is not specified (option Number of attributes... left unchecked), this number is equal to the square root of the number of attributes in the data.  Replicable training: Fix the seed for tree generation, which enables the replicability of the results.  Balance class distribution: [Weigh classes](https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html?highlight=sklearn%20utils%20class_weight) inversely proportional to their frequencies. |
| Growth control | Limit depth of individual trees: Original Breiman proposes to grow the trees without any pre-pruning, but since pre-pruning often works quite well and is faster, the user can set the depth to which the trees will be grown.  Do not split subsets smaller than, select the smallest subset that can be split. |

## Information about test and training datasets



### Number of data objects in the training dataset: **275**

### % proportion of data objects in the training dataset: **%70**

| **Class label** | **Number of data objects in the training dataset** | **% proportion of data objects in the training dataset** |
| --- | --- | --- |
| usa | 175 | %63.64 |
| japan | 53 | %19.27 |
| europe | 47 | %17.09 |

### Number of data objects in the test dataset: **117**

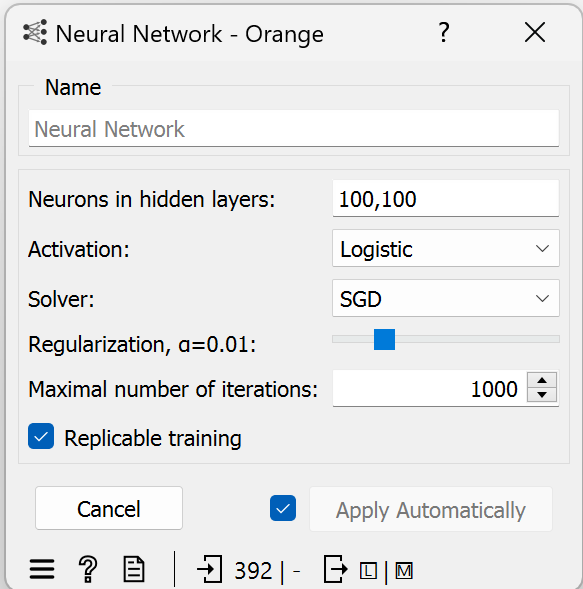
### % proportion of data objects in the test dataset: **%30**

| **Class label** | **Number of data objects in the test dataset** | **% proportion of data objects in the test dataset** |
| --- | --- | --- |
| usa | 70 | %59.83 |
| japan | 26 | %22.22 |
| europe | 21 | %17.95 |

## Experiments with artificial neural network

| **Experiment** | **Hyperparameter values** |
| --- | --- |
| Experiment 1 | Neurons in hidden layers: 100,100  Activation: Logistic  Solver: SGD  Regularisation: alpha=0.01  Max. Iterations: 1000 |
| Experiment 2 | Neurons in hidden layers: 100,100, 100  Activation: Logistic  Solver: SGD  Regularisation: alpha=0.01  Max. Iterations: 1000 |
| Experiment 3 | Neurons in hidden layers: 100,50, 50  Activation: Logistic  Solver: SGD  Regularisation: alpha=0.01  Max. Iterations: 1000 |

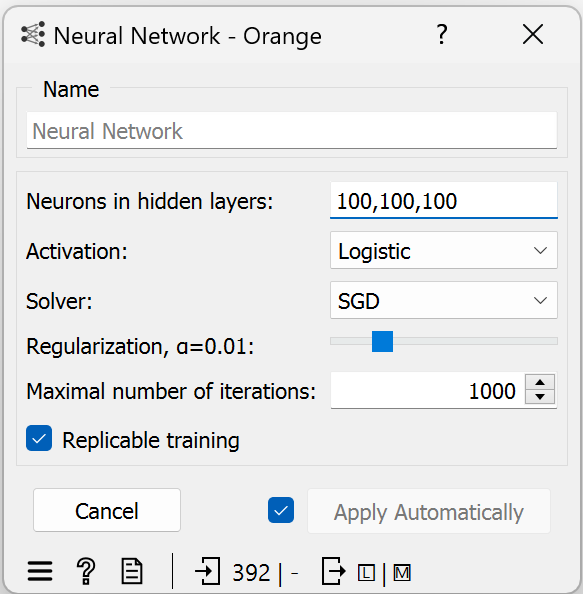
<a screenshot of hyperparameter values for Experiment 1>

****

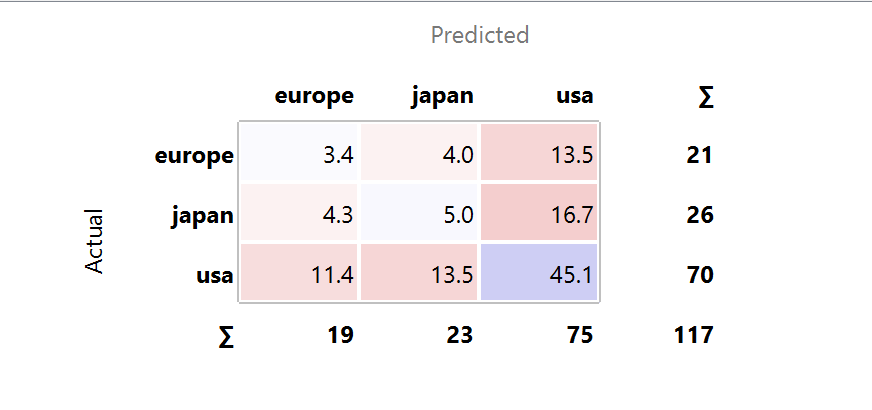
<a screenshot of performance metrics for Experiment 1>

****

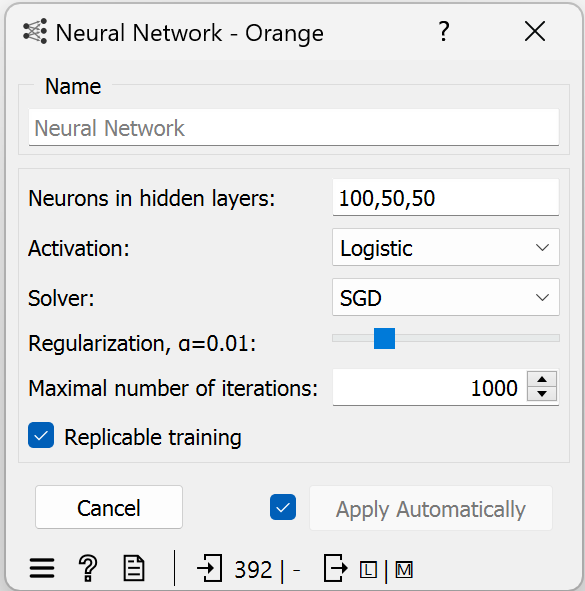
<a screenshot of hyperparameter values for Experiment 2>

****

<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:*

All three experiments investigating an automobile dataset with neural networks exhibited signs of underfitting. Despite trying different hidden layer configurations (100,100; 100,100,100; 100,50,50) and keeping the same activation function (logistic), solver (SGD), regularisation (L1 alpha=0.01), and training iterations (1000), the models consistently underpredicted values across all categories (Europe, Japan, USA) in the test set. So in this experiment, we basically tried to see the effects of ‘Neurons In Hidden Layers’ property changes. We saw that adding another layer ended up with us getting more accurate values while inputting higher values to the layer caused our values to go higher as well.

### Model selected for testing:

For the testing process, Experiment 2 with three hidden layers of 100 neurons each will be selected as the model of choice. This decision is based on the overall consistency in performance observed across experiments, as Experiment 2 exhibited similar accuracy levels and stability compared to Experiment 1 and Experiment 3. Additionally, selecting Experiment 2 provides a balanced approach with a moderate level of network complexity, making it a suitable candidate for further testing and evaluation.

Neurons in hidden layers: 100,100, 100

Activation: Logistic

Solver: SGD

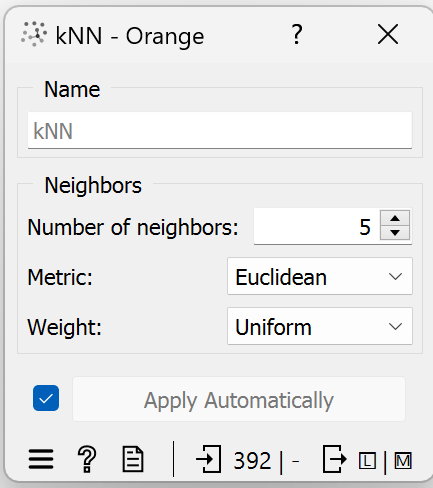
Regularisation: alpha=0.01

Max. Iterations: 1000

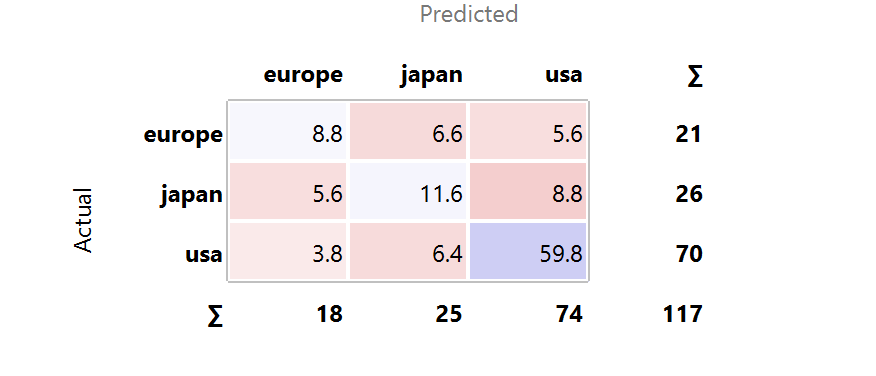
## Experiments with kNN

| **Experiment** | **Hyperparameter values** |
| --- | --- |
| Experiment 1 | Number of neighbours:5  Metric: Euclidean  Weight: Uniform |
| Experiment 2 | Number of neighbours:10  Metric: Euclidean  Weight: Uniform |
| Experiment 3 | Number of neighbours:20  Metric: Euclidean  Weight: Uniform |

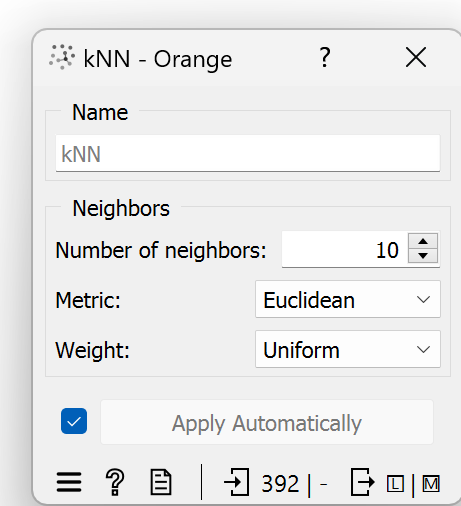
<a screenshot of hyperparameter values for Experiment 1>

****

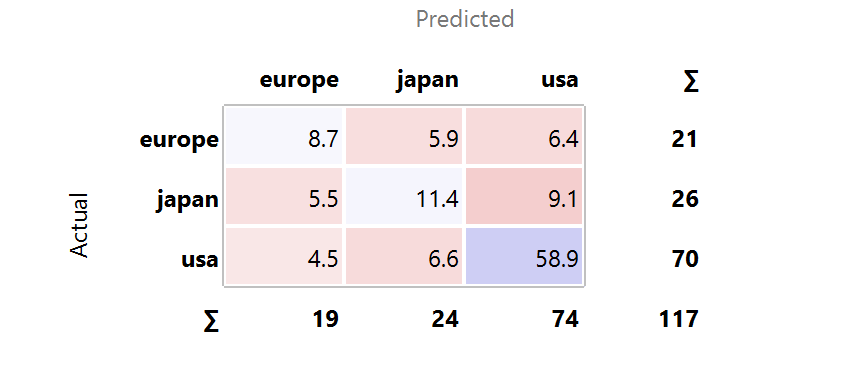
<a screenshot of performance metrics for Experiment 1>

****

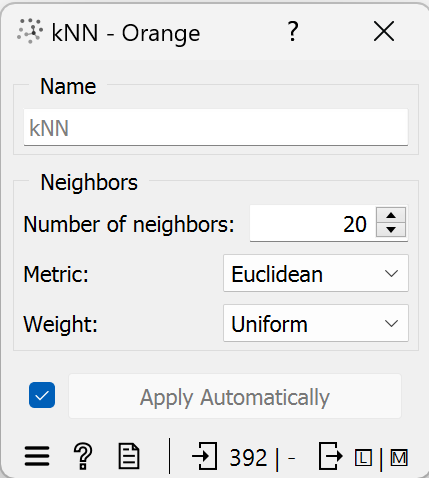
<a screenshot of hyperparameter values for Experiment 2>

****

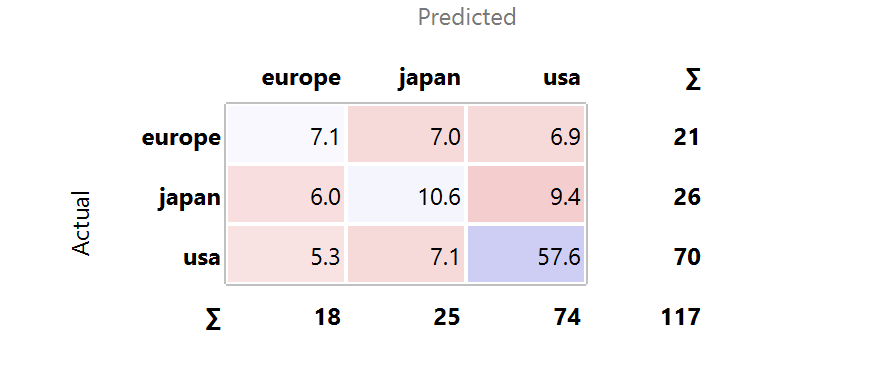
<a screenshot of performance metrics for Experiment 2>

****

<a screenshot of hyperparameter values for Experiment 3>

****

<a screenshot of performance metrics for Experiment 3>

****

*Conclusions from experiments:*

The experiments conducted with kNN (k-Nearest Neighbors) classification provide valuable insights into the performance of the algorithm under various configurations. Across the three experiments, where the number of neighbors varied from 5 to 20, consistent trends in classification accuracy and class distribution were observed. In Experiment 1, utilizing 5 neighbors, the classifier demonstrated high accuracy, particularly for the majority class (USA), showcasing its effectiveness in correctly classifying instances. This trend continued in Experiment 2 with 10 neighbors, where slight improvements in accuracy were noted, albeit with minimal changes in classification patterns. Experiment 3, employing 20 neighbors, exhibited similar performance trends, indicating that increasing the number of neighbors beyond a certain threshold did not substantially impact classification outcomes.

Throughout all experiments, the choice of the Euclidean distance metric and uniform weight scheme provided consistent classification results. These settings facilitated robust performance across different configurations, demonstrating the reliability of the kNN algorithm in handling the classification task. However, despite the overall high accuracy, slight fluctuations in the classification accuracy of minority classes (Europe and Japan) were observed across experiments. This suggests potential challenges in accurately distinguishing between these classes, likely due to their imbalanced distribution within the dataset.

### Model selected for testing:

For the testing process, Experiment 2 with 10 neighbors, employing the Euclidean distance metric and uniform weight scheme, will be selected as the model of choice. This decision is based on the slight improvements in classification accuracy observed in Experiment 2 compared to Experiment 1, while maintaining consistency in performance across different configurations. Additionally, Experiment 2 provides a balanced trade-off between model complexity and classification accuracy, making it a suitable candidate for further testing and evaluation.

Number of neighbours:10

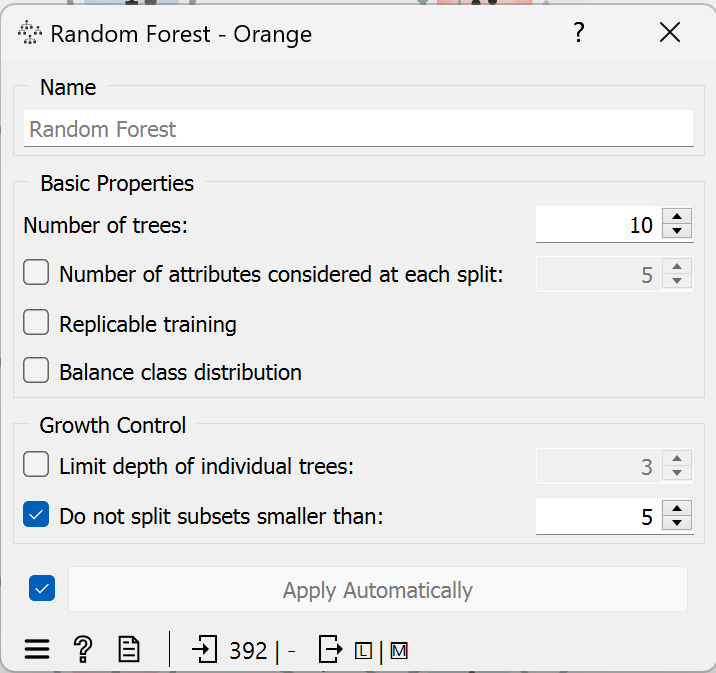
Metric: Euclidean

Weight: Uniform

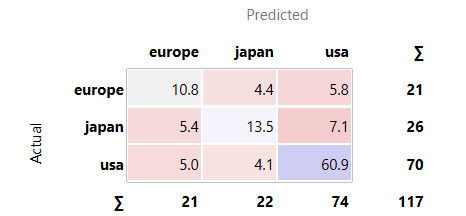
## Experiments with Random Forest

| **Experiment** | **Hyperparameter values** |
| --- | --- |
| Experiment 1 | Number of trees:10  Do not split subsets smaller than:5 |
| Experiment 2 | Number of trees:15  Do not split subsets smaller than:5 |
| Experiment 3 | Number of trees:20  Do not split subsets smaller than:5 |

<a screenshot of hyperparameter values for Experiment 1>

****

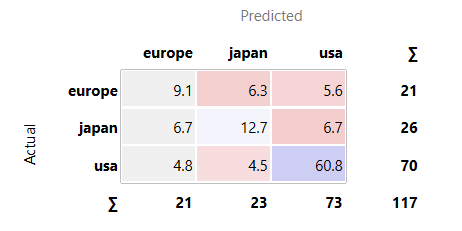
<a screenshot of performance metrics for Experiment 1>

****

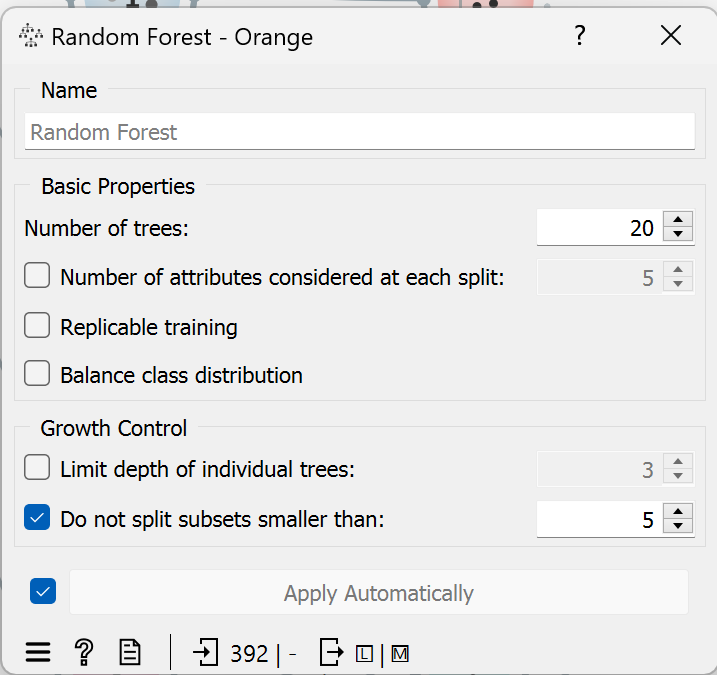
<a screenshot of hyperparameter values for Experiment 2>

****

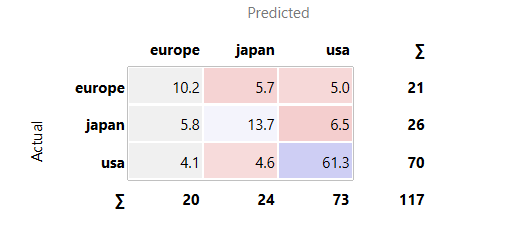
<a screenshot of performance metrics for Experiment 2>



<a screenshot of hyperparameter values for Experiment 3>

****

<a screenshot of performance metrics for Experiment 3>



*Conclusions from experiments:*

The experiments conducted with Random Forest classification offer valuable insights into the performance of the models under various configurations. Across all experiments, which varied the number of trees in the ensemble from 10 to 20, consistent trends in classification accuracy and class distribution were observed. Surprisingly, altering the number of trees did not yield significant differences in model performance, suggesting that the Random Forest algorithm maintained stability and robustness regardless of the ensemble size. Despite this consistency, there were minor fluctuations in the accuracy of class predictions, particularly for minority classes such as Europe and Japan. However, the majority class (USA) consistently exhibited the highest accuracy, highlighting potential challenges posed by imbalanced class distributions within the dataset.

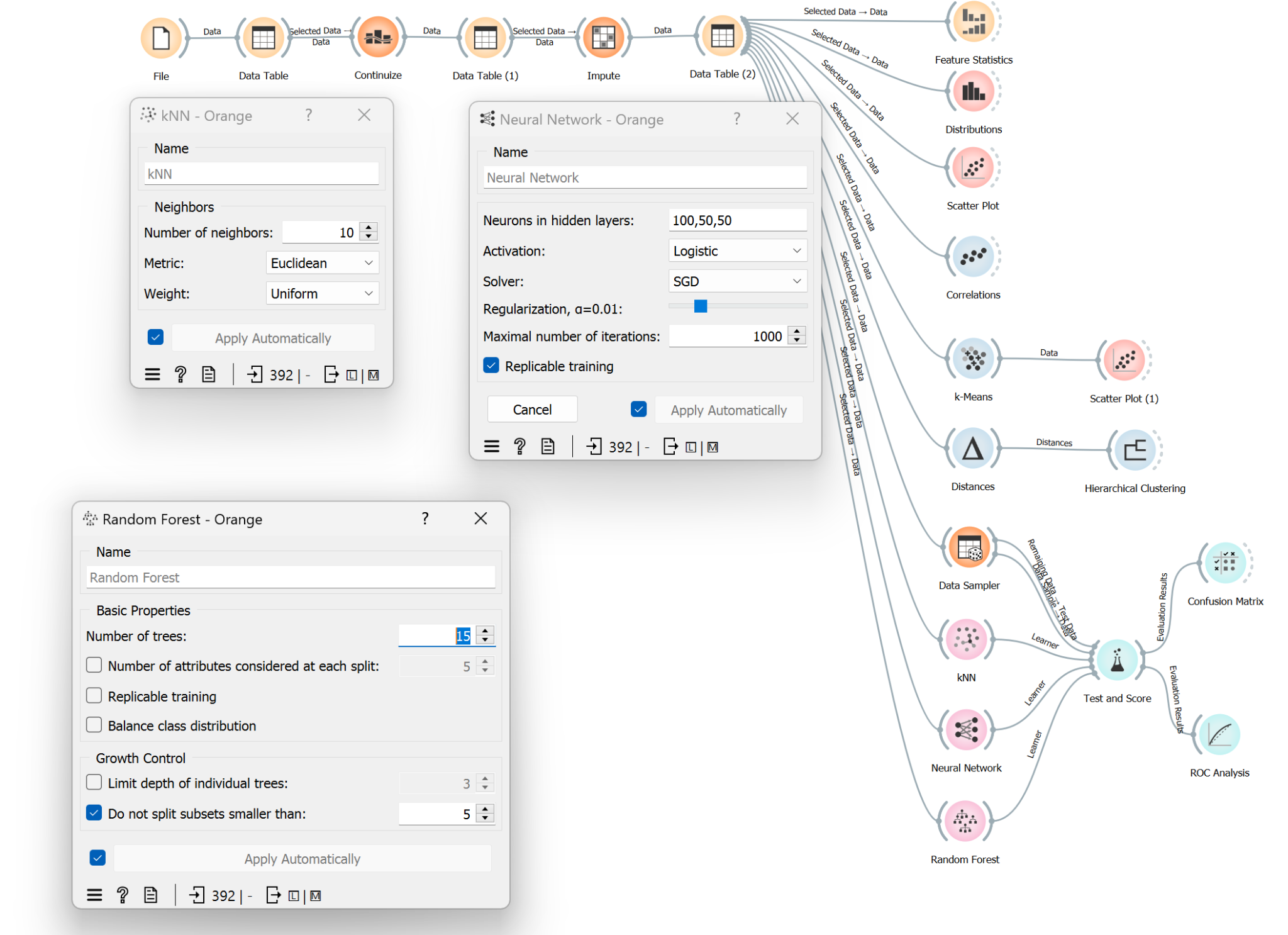
### Model selected for testing:

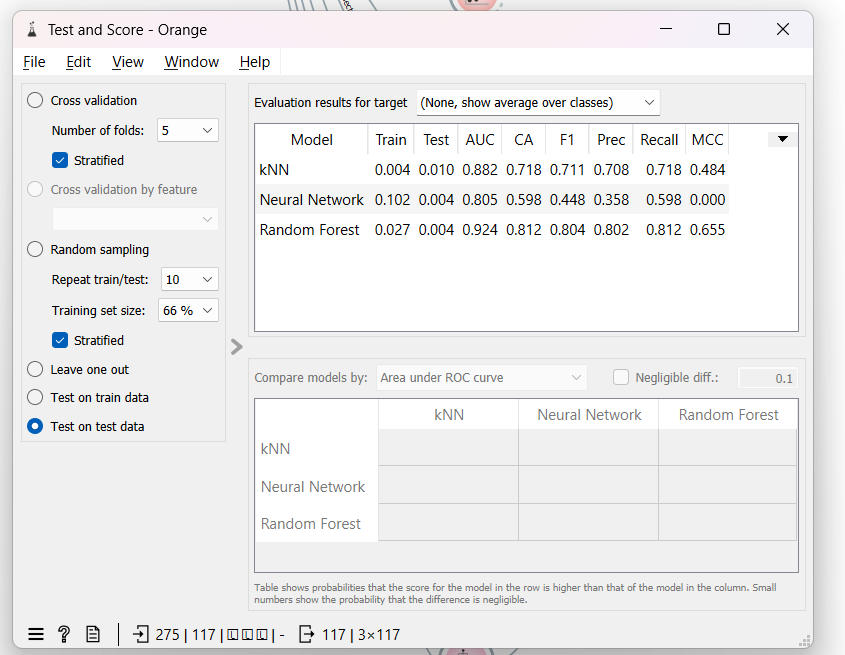
For the testing process, Experiment 2 with 15 trees in the Random Forest ensemble will be selected as the model of choice. This decision is based on the consistency of performance observed across experiments, where Experiment 2 exhibited similar accuracy levels and stability compared to Experiment 1 and Experiment 3. Additionally, selecting Experiment 2 provides a balanced approach, as it represents a middle ground in terms of the number of trees, offering a reasonable compromise between model complexity and classification accuracy.

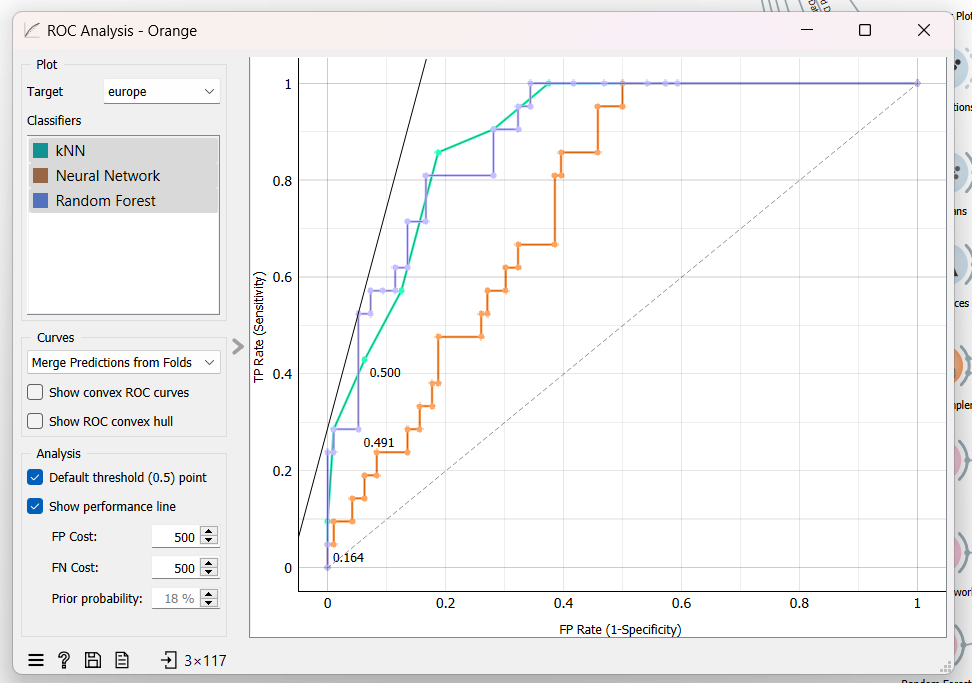
Number of trees:15

Do not split subsets smaller than:5

## Testing results of the trained models



**

**

*Conclusions after testing:*

In our various experiments, we consistently aimed to keep our values as close to the predictions as possible, with efficiency being our top priority. Based on the results we obtained, we integrated all of our selected experiment hyperparameters into our workflow. The fact that the data we obtained in the tables above is more accurate than the results of our experiments is evidence that we made the correct choices.

According to the test and score metrics, our best working algorithm is Random Forest since AUC value is the highest with 0.924, we have the lowest value of 0.805 with Neural Network. We can also see that Random Forest took more process time than the others.

Also, the F1 score is the harmonic mean of precision and recall and provides a balance between the two metrics. With that information, we can make the conclusion that Random Forest has the F1 score of 0.804, kNN has 0.711 and Neural Network has 0.448. With that being said, we can clearly say that Random Forest has the best precision among all.

# Information sources

[Kaggle](https://www.kaggle.com/datasets/yasserh/wine-quality-dataset/discussion/308022)

[YouTube](https://www.youtube.com/channel/UClKKWBe2SCAEyv7ZNGhIe4g)

[Orange](https://orangedatamining.com/getting-started/)

[ORTUS](https://estudijas.rtu.lv/course/view.php?id=361245&section=10#h5pbookid=21751&section=top&chapter=h5p-interactive-book-chapter-99cd44ec-617a-4c7a-a397-98ef4dfb8b30)

[Hierarchical Clustering in Machine Learning - Javatpoint](https://www.javatpoint.com/hierarchical-clustering-in-machine-learning#:~:text=The%20agglomerative%20hierarchical%20clustering%20algorithm%20is%20a%20popular%20example%20of,closest%20pair%20of%20clusters%20together)