# Consumer Study Data Analysis

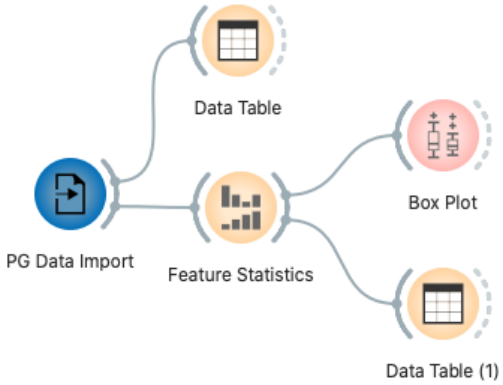
## Working notes for the hands-on course at Procter & Gamble

In this tutorial, we will work with the Consumer Study data. We will make the basic overview of the data, observe correlations between attributes and perform clustering analysis on the attributes (consumer study questions). At the end of the lesson, we will also touch on predictive analysis. With regression models, we will try to predict Overall rating from other attributes and explain model predictions.

Handouts prepared by:
Bruna Zupan, Lan Žagar and Primož Godec

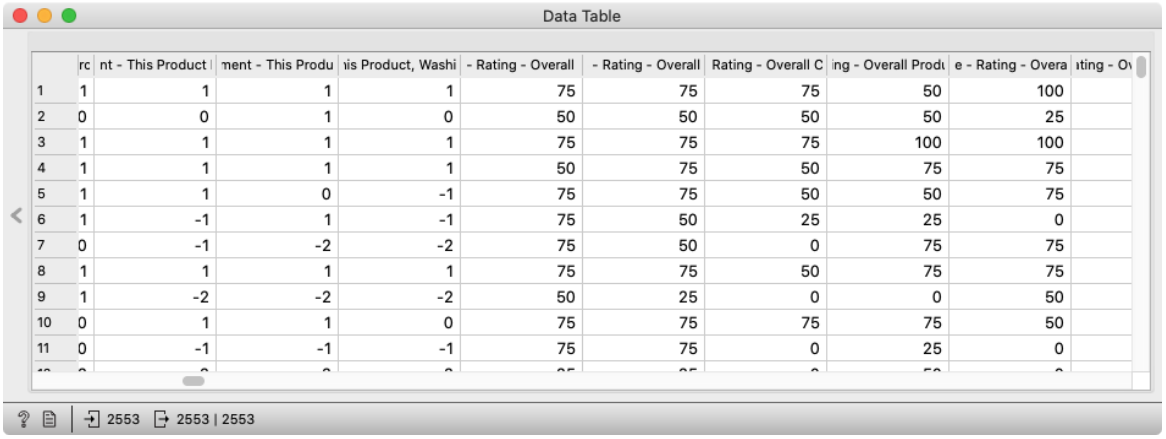
# Lesson 1: Data overview

Before we start with the analysis, we would like to get some insight into the data. We will use the *Data Table*, which shows data in tabular form, and the *Feature Statistics*, which computes simple statistics about each attribute in the data.



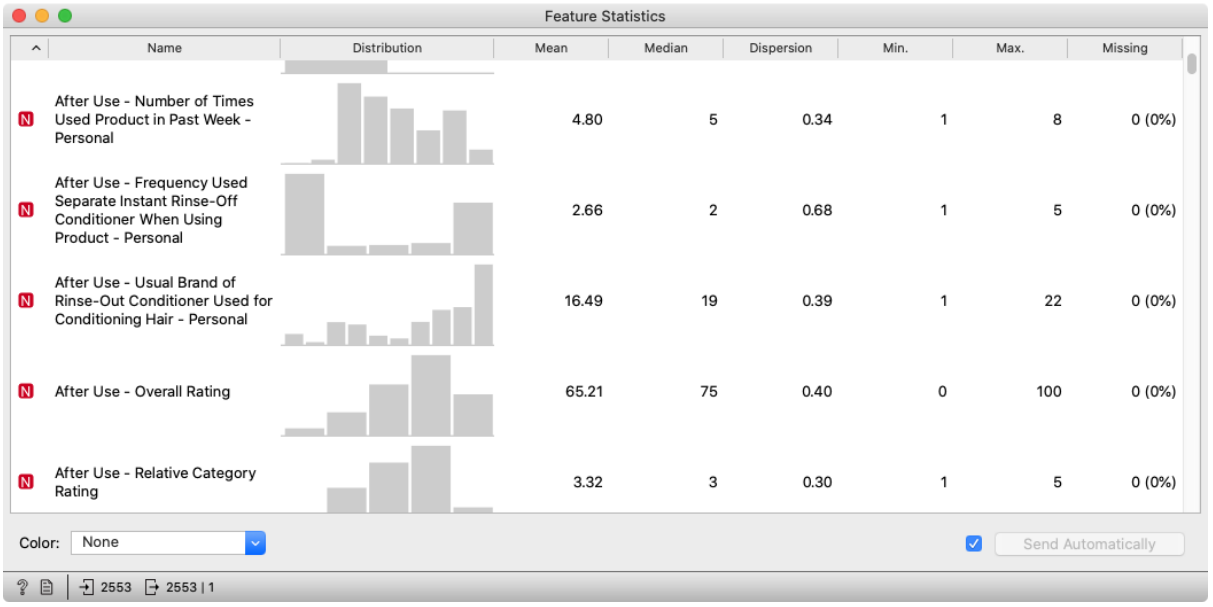

First, we load consumer study data with the *PG Data Import*. To get a basic overview of values and attributes, we connect the *Data Table* to *PG Data Import* and make sure that we use the Consumer Study output from the *PG Data Import* with a double click on the connection between widgets.

We open the *Data Table* and observe the data in the tabular view (view similar to Excel spreadsheet). Each line in the table is a data instance — response in the consumer study, and each column is an attribute. Most of the attributes represent answers from the consumer study. At the bottom of the widget, we can see the number of input instances — the number of responses in the consumer study, and some other insights in data.




| | rc | nt - This Product | ment - This Produ | is Product, Washi | - Rating - Overall | - Rating - Overall | Rating - Overall C | ing - Overall Prodı | e - Rating - Overa | ıting - Ov |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 75 | 75 | 75 | 50 | 100 | |
| 2 | 0 | 0 | 1 | 0 | 50 | 50 | 50 | 50 | 25 | |
| 3 | 1 | 1 | 1 | 1 | 75 | 75 | 75 | 100 | 100 | |
| 4 | 1 | 1 | 1 | 1 | 50 | 75 | 50 | 75 | 75 | |
| 5 | 1 | 1 | 0 | -1 | 75 | 75 | 50 | 50 | 75 | |
| 6 | 1 | -1 | 1 | -1 | 75 | 50 | 25 | 25 | 0 | |
| 7 | 0 | -1 | -2 | -2 | 75 | 50 | 0 | 75 | 75 | |
| 8 | 1 | 1 | 1 | 1 | 75 | 75 | 50 | 75 | 75 | |
| 9 | 1 | -2 | -2 | -2 | 50 | 25 | 0 | 0 | 50 | |
| 10 | 0 | 1 | 1 | 0 | 75 | 75 | 75 | 75 | 50 | |
| 11 | 0 | -1 | -1 | -1 | 75 | 75 | 0 | 25 | 0 | |

The second widget that we connect to the PG Data Import is *Feature Statistics*. It gives some basic information about attributes such as the average value, number of missing values, and visualizes the distribution of answers.

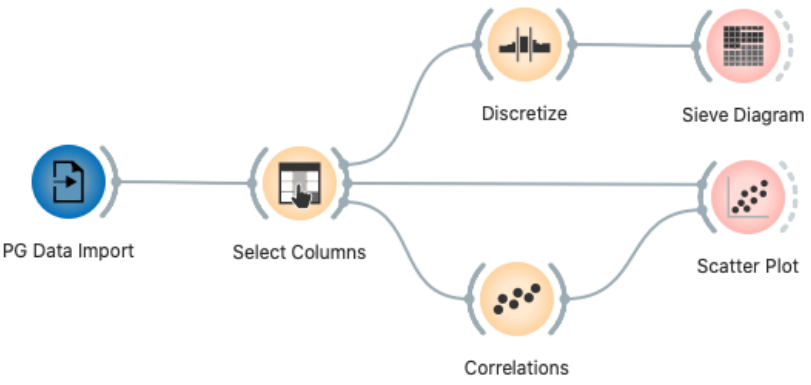



In the figure, we can observe that the attribute `After Use -Frequency Used Separate Instant Rinse-Off Conditioner When Using Product- Personal` include values from 1 (min column) to 5 (max), its average response is 2.66, and it has no missing values. From the graph, we can also see that participants mostly selected-response 1 and response 5.

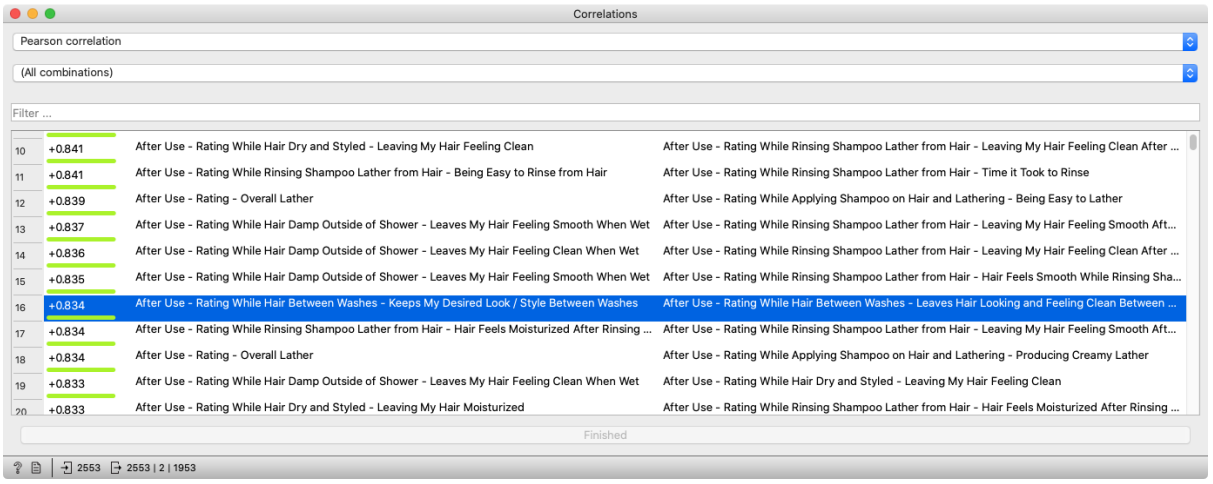
In the *Feature Statistics*, we can select attributes (rows) that we are interested in the most and use them in further analysis. For the demonstration, we connected *Box Plot* to the output of the *Feature Statistics* and observed selected attributes there.

# Lesson 2: Correlations

Now when we know our data, we can start with the analysis. First, we try to find attributes that correlate with each other.



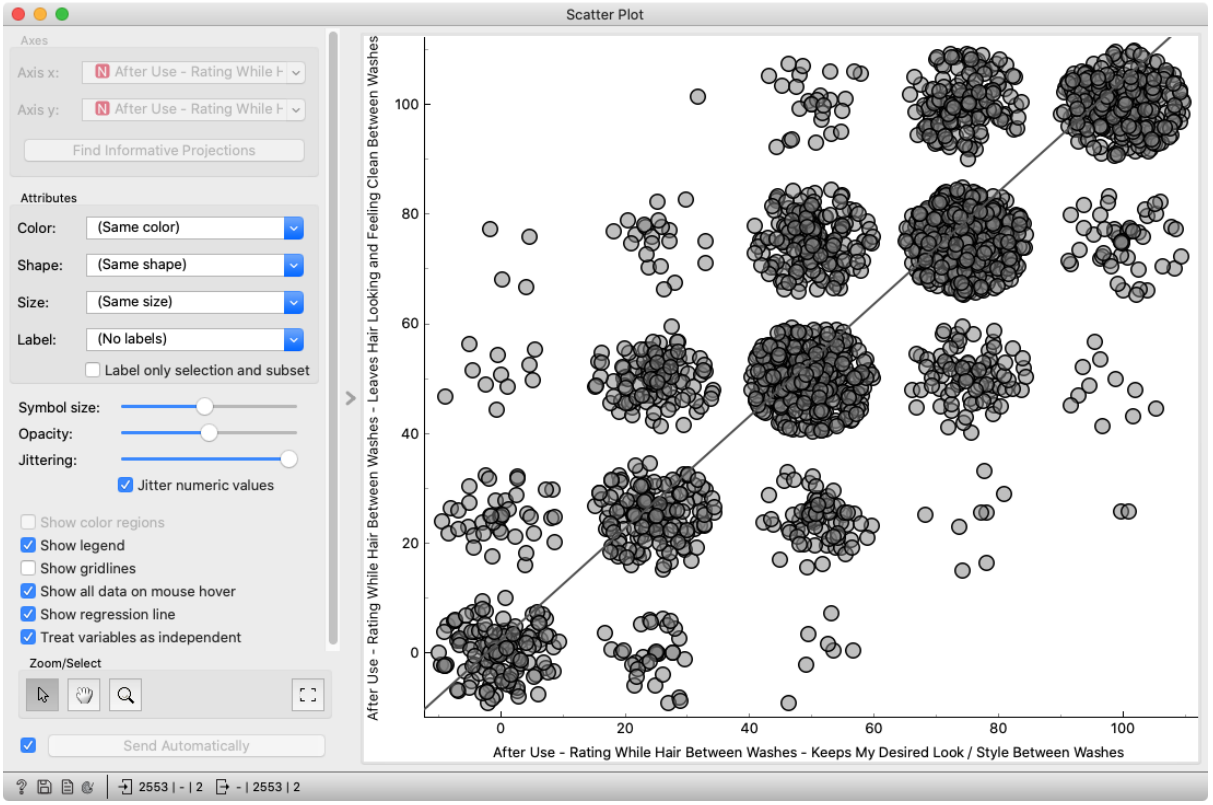

We load data with the *PG Data Import* widget as before. In the *Select Columns* widget, we select a manageable number of attributes. We decided to analyze all features from the *After use - Rating group*.
In the *Correlation*, we observe attributes that correlate with each other and how. Combinations of attributes are sorted depending on Pearson correlation.




Correlation (in the first column) is the number between -1 and 1, where both extremes show correlation and 0 means there is no correlation. Values larger than 0 indicate positive correlations — if the value of the first attribute increases, the second attribute's corresponding value should increase too. The negative correlation means that while the value of the first attribute increases, the second decreases.

In data, we can observe a strong correlation between some attributes that measure the similar property of the shampoo, but also some not so obviously connected attributes.

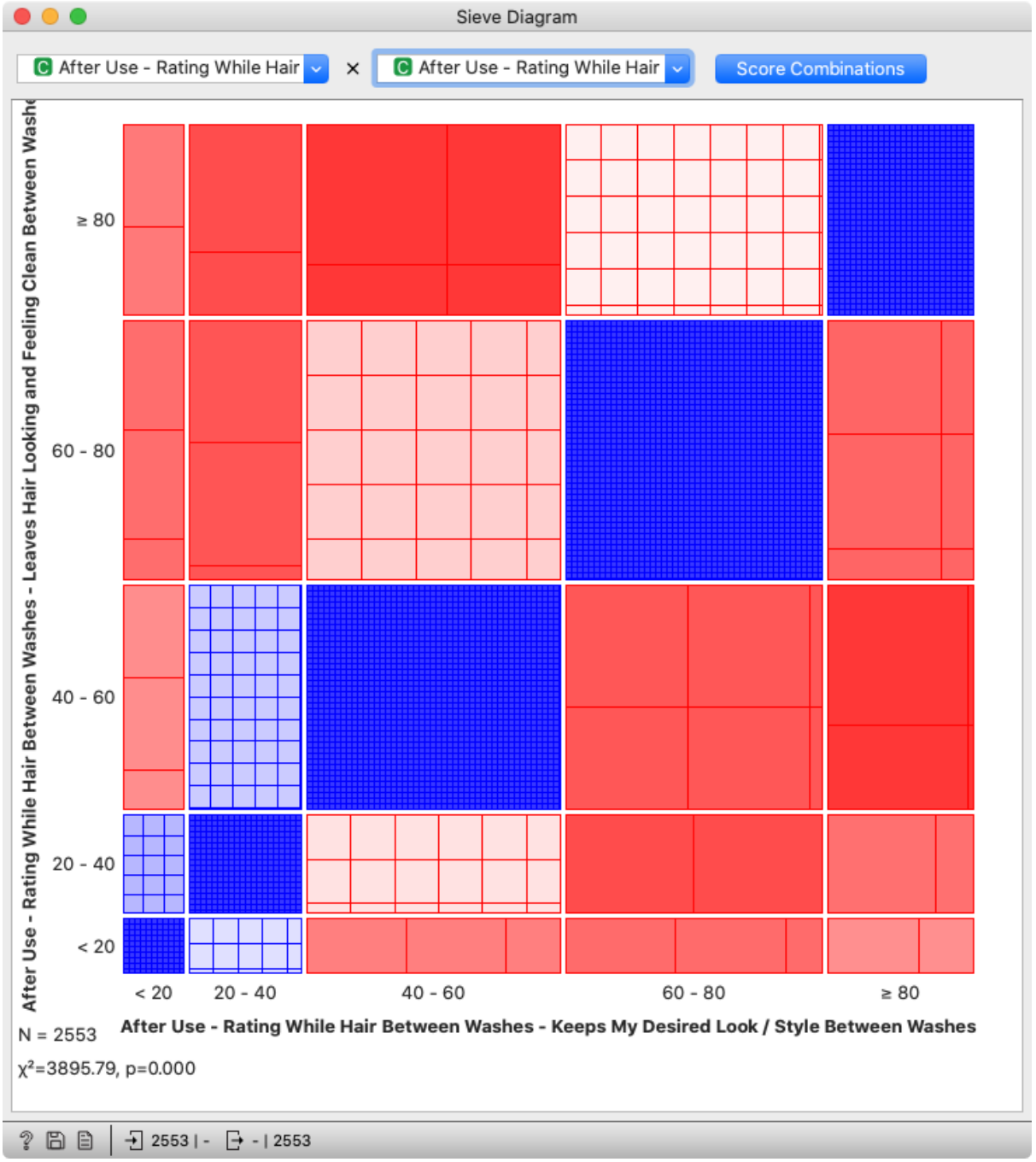
In the widget, we select a combination of correlated attributes that look interesting. The Feature output of the Correlation widget is connected to the *Scatter Plot*. It tells the *Scatter Plot* to show the combination of selected two attributes.




Since selected attributes are ordinal (consist of values 0, 25, 50, 75, 100), many points appear at the same position on the plot. Thus we apply some Jittering to disperse points. We can see that the selected two attributes indeed correlate since most of the values follow the line plotted in the graph. The majority of answers have the same response to both of the selected questions in the consumer study.

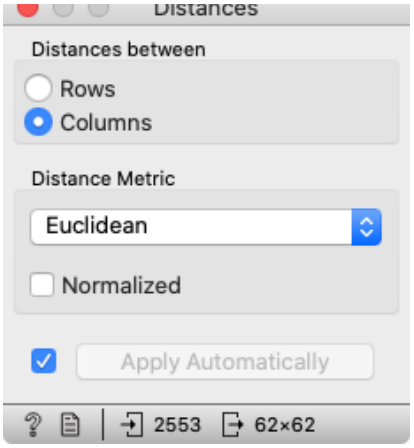
We can see the correlation in the scatter plot, but we can see a detailed picture of correlation with the *Sieve Diagram*. Each rectangle area is proportional to the expected frequency, while the observed frequency is shown by the number of squares in each rectangle. The difference between observed and expected frequency appears as the shading density, using color to indicate whether the deviation from independence is positive (blue) or negative (red).

As expected, combinations on the diagonal have more observations than expected. Most other combinations have fewer observations than expected. Anyway, we can see that participants who selected lower ratings (0, 25) for `Keep My Desired Look/Style Between Washes` more frequently decided to give slightly better ratings for the question on Y-axis.



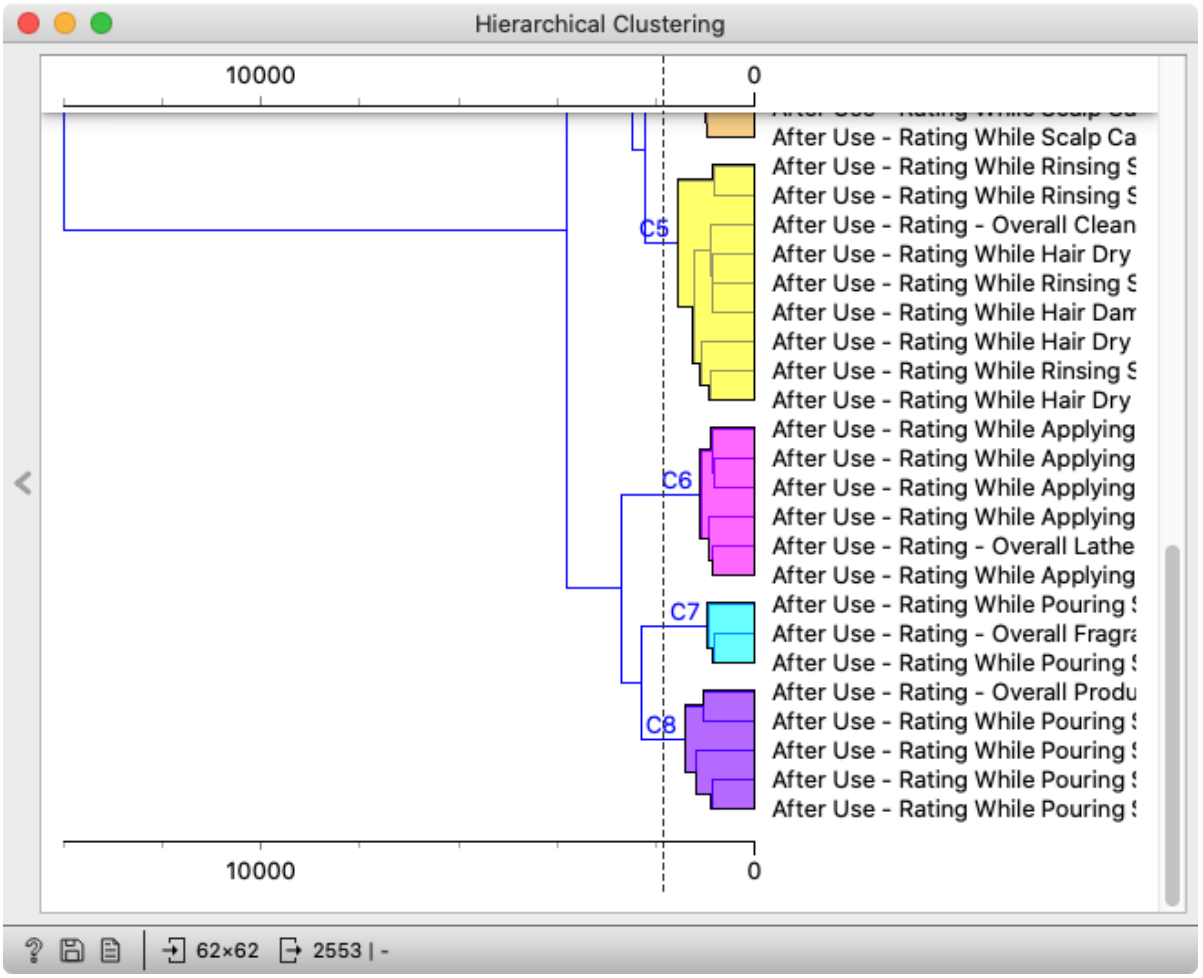
# Lesson 3: Question clustering


In the previous lesson, we were correlating attributes — questions from the consumer study. In this lesson, we will go a step further and try to identify clusters of similar attributes.




As before, we load data and select a manageable subset of attributes. We use the *Distances* widget to compute the distance between attributes. In the widget, we select to compute distances between columns (attributes).

We use *Hierarchical Clustering* to compute hierarchical clustering on the matrix of distances and show the dendrogram. In the widget, we select *Ward* linkage and feature names as annotations.

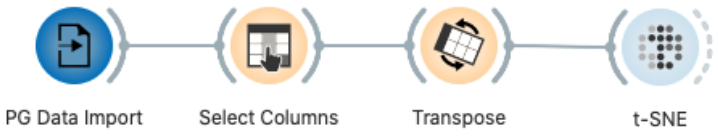


The dendrogram shows groups of similar attributes (answers) based on responses from participants. The length of the connection shows how similar response vectors are. A closer length of connection means a smaller distance between groups/attributes.

In the plot, we can see some obvious groups of attributes since they ask for the rating of the similar property of shampoo, but some are not so obvious. For example, in group C8, attributes `Overall Product Appearance` is closely related to the `Color of Shampoo`, which is obvious, but also to dispensing property.

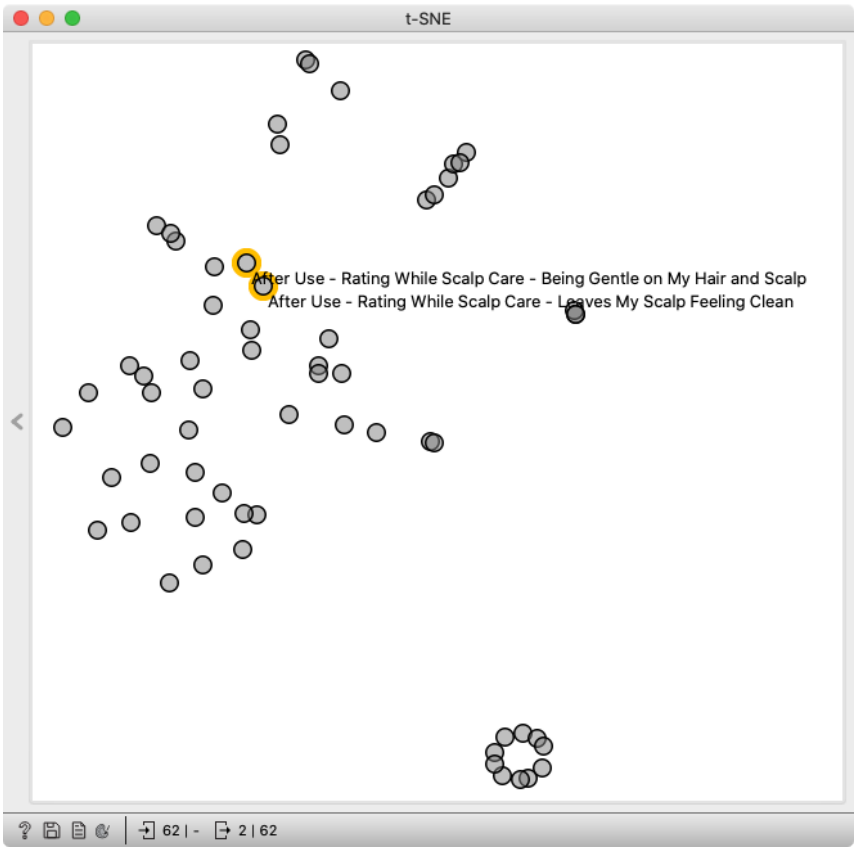
# Lesson 4: Representative cluster attributes

In the previous lesson, we used the hierarchical clustering of attributes on the dataset. Now we will analyze attribute similarities (based on responses) on a two-dimensional plane using t-SNE.

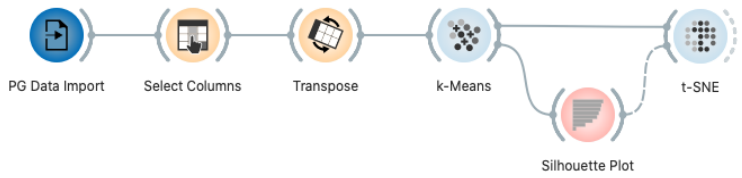



Again we load consumer study data and select the same attributes as before. The *Distance* widget that we have used before is a bit special since it allows us to compute distances between columns. In this case, *t-SNE* does not have an option to map columns (attributes), so we need to transpose the dataset before.

*t-SNE* maps data instances (now each instance is an attribute from the original table) on the two-dimensional plane, based on their similarity. Similar attributes will be close on the plane.

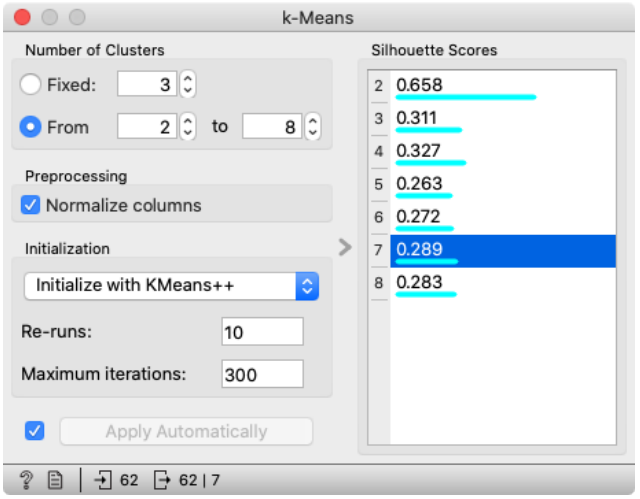


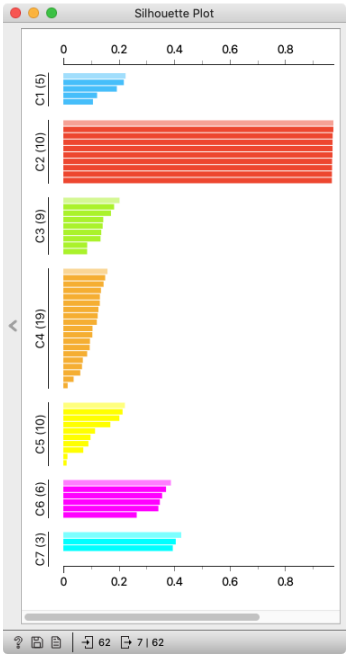
In the image, we can observe a similar group of attributes. We can select them and see their names or even observe them in the *Data Table* connected to the output of the widget.




On the *t-SNE* map, we can observe a similar group of attributes. We can select them and see their names or even explore them in the *Data Table*.

Now when we have the *t-SNE* mapping, we can enhance it with clustering. We will use *k-Means* clustering to find clusters in the data and use them in the *t-SNE* plot. For clustering, we insert the *k-Means* widget between *Transpose* and *t-SNE*. In the *k-Means* widget, we set the number of clusters to 7.



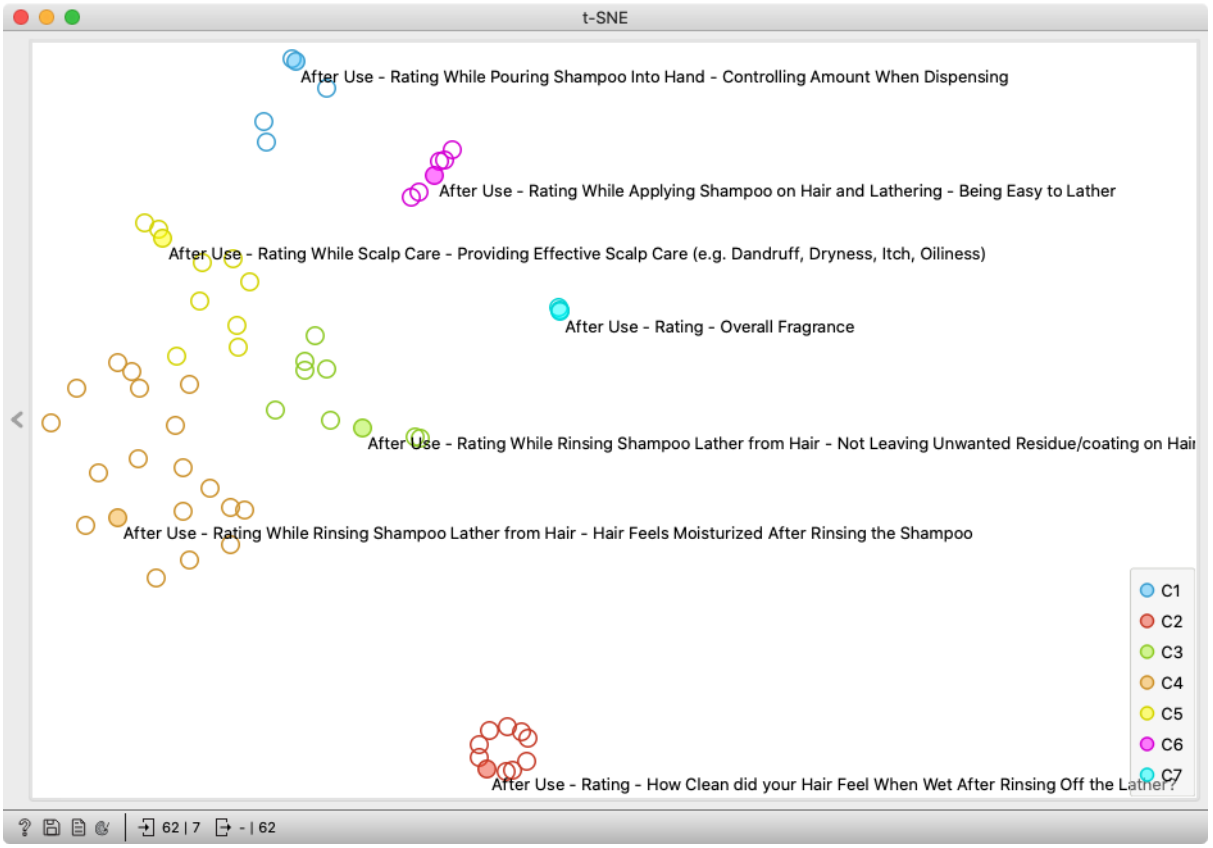




We connect the *Silhouette Plot* to the *k-Means*, which offers a graphical representation of consistency within clusters of data and provides the user with the means to visually assess cluster quality. The silhouette score is a measure of how similar an object is to its cluster in comparison to other clusters. The silhouette score close to 1 indicates that the data instance is close to the center of the cluster and instances possessing the silhouette scores close to 0 are on the border between two clusters.

In the *Silhouette Plot*, we select the best scoring instance in each cluster. Those are instances closest to the center of the cluster, so we believe that they best represent the attributes of clusters.

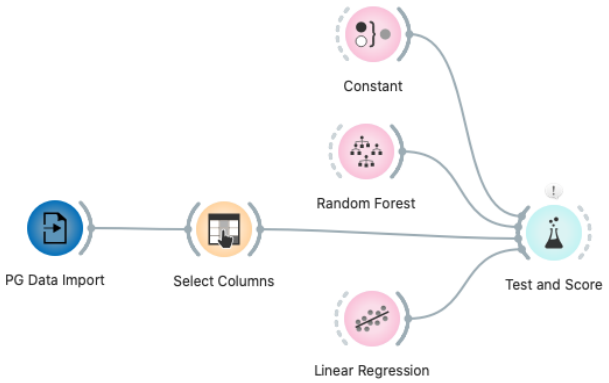
We connect the Selected data output of the *Silhouette Plot* to the Data Subset of the *t-SNE* widget. Selected instances are plotted with the filled circle in the *t-SNE,* and if we set the Label attribute to

Feature name, labels will be present beside select instances. Select attributes are attributes that represent the center of the cluster.



# Lesson 5: Predicting overall rating

Now we will try something different. We will predict the overall score the user gave to the product based on other rationings in the data and observe what attributes contribute the most to the prediction.




We again load the data and select all features from the `After use - Rating` group as Features and `After use - Overall rating as a Target` attribute.

We use *Linear Regression* and *Random Forest* (regression model) as predictors. *Constant* — a model which predicts an average value for each instance — is used as a baseline model to see if used models make meaningful predictions. For testing the models' accuracy, we use the *Test and Score* widget.

Test and Score

Evaluation Results

| Model | MSE | RMSE | MAE | R2 |
|---|---|---|---|---|
| Random Forest | 228.347 | 15.111 | 11.493 | 0.668 |
| Linear Regression | 226.533 | 15.051 | 11.455 | 0.671 |
| Constant | 688.591 | 26.241 | 21.827 | -0.001 |

Model Comparison by MSE

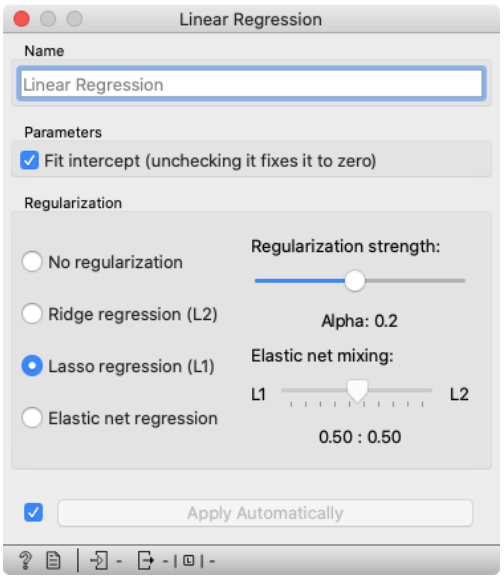
| | Random Forest | Linear Regression | Constant |
|---|---|---|---|
| Random Forest | | 0.591 | 0.000 |
| Linear Regression | 0.409 | | 0.000 |
| Constant | 1.000 | 1.000 | |

Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.

2553 | - | 2553 | 3×2553 Stratification is ignored for regre...

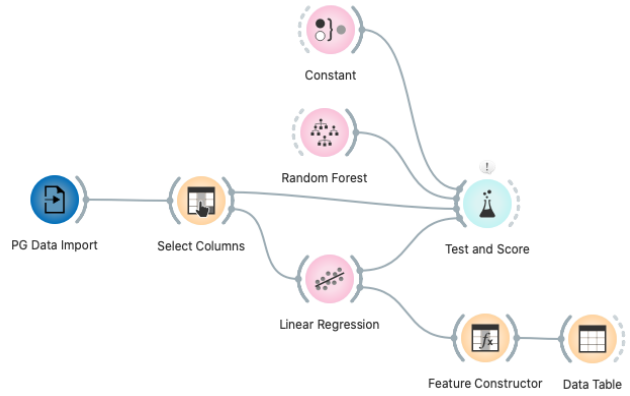
We performed 5-fold cross-validation. In the result table, we can see that both model's predictions — with lower MAE (Mean absolute error) -- are significantly better than an average prediction. MAE score of 11.45 tells us that averagely model mistakes for 11.45 compared to a true value.

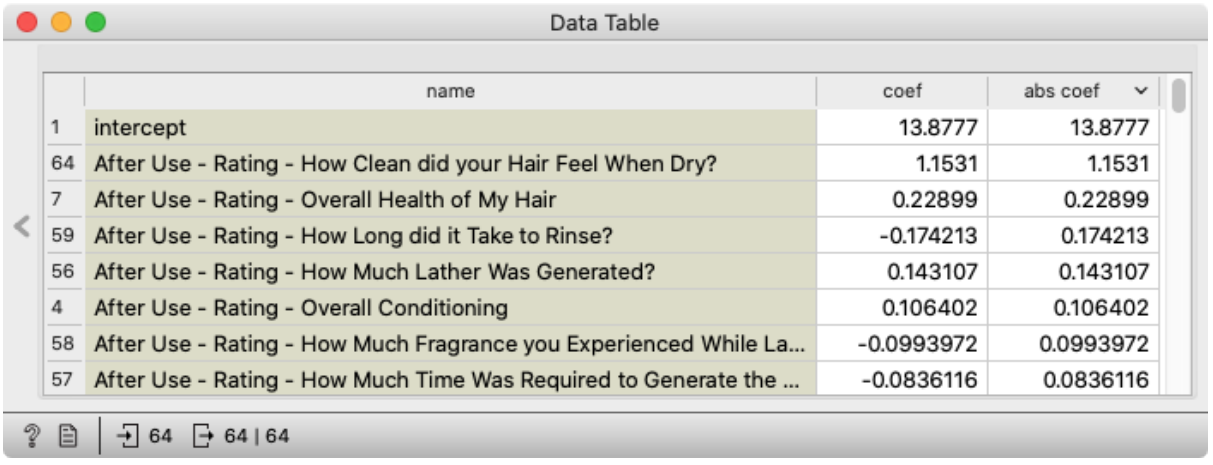
# Lesson 6: Explaining predictions



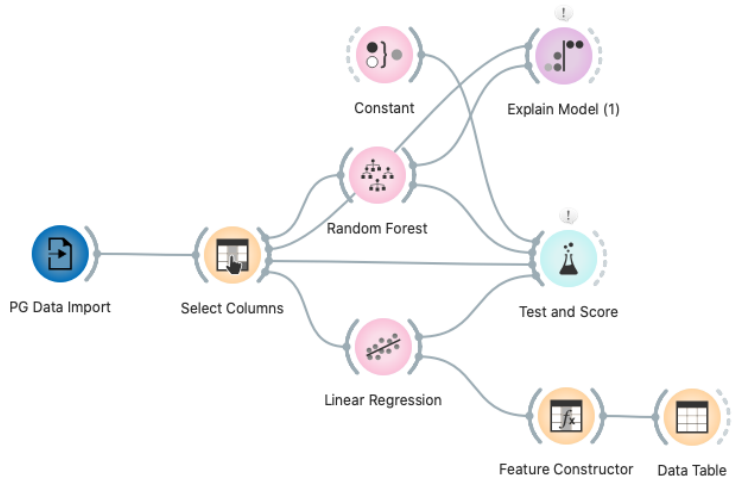

In the previous lesson, we observed that the best scoring method is *Linear regression*. Since we wanted to avoid overfitting and to get insight into the prediction — we wanted to know what features contribute the most to the prediction — we used *Lasso (L1) regularization* with regularization strength 0.2.

In this lesson, we connect input data to the *Linear Regression* widget such that we train the model independently of the *Test and Score* widget. We use the *Data Table* and connect it to *Linear Regression*'s coefficients output to see the model's coefficients. In between, we use the feature constructor widget to get absolute values of the coefficients. Since we use Lasso regularization, which forces coefficients for important features to have high values and others towards zero, we can explain the feature importance towards the predictions. Features with higher values are more important for predicting the output class — in our case `Overall rating`. This way, we can see which features contribute the most towards participants' decision toward `Overall rating`.



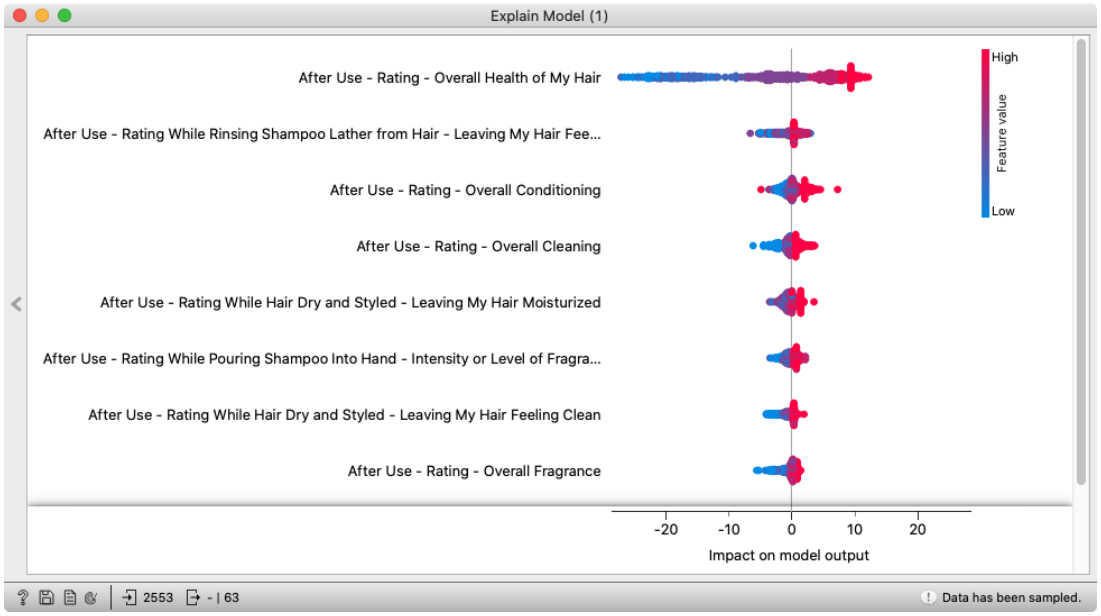

In the table, with the coefficients, we can see that the most important factor for the Overall score prediction is how to clean participants' hair




when they are dry. It seems that overall health and time taken to rinse are important too.

Since linear regression is a simple model, we can easily explain prediction by observing the coefficients. It is not the case for the random forest model, which is more complex. To explain its predictions, we can use the *Explain Model* widget.

Feature order in the graph indicates feature importance for the prediction. It also gives information about how features contribute. Blue colors represent instances with low values of the attribute, and red colors represent the high values of the attribute. On the graph, they indicate whether they contribute positively or negatively. The blue value on the left part of the graph indicates that the low value of the attribute decreases the predicted value (predicted Overall score for this participant).




In the graph, we can see that the most important feature is the overall health of the hair. A high score for this feature increases the final prediction of the Overall score, while a low value contributes toward decreasing the predicted value. The second most important attribute is how clean the hair is after rinsing and it contributes in the same direction.