EuroVis Workshop on Visual Analytics (2022)
M. Angelini and J. Bernard (Editors)

RankASco: A Visual Analytics Approach to Leverage Attribute-Based User Preferences for Item Rankings Supplemental Material

J. Schmid[1], L. Cibulski[2,3], I. Al Hazwani[1] and J. Bernard[1,4]

[1]University of Zurich, Switzerland
[2]Fraunhofer IGD, Darmstadt, Germany
[3]Technical University of Darmstadt, Darmstadt, Germany
[4]Digital Society Initiative, Zürich, Switzerland

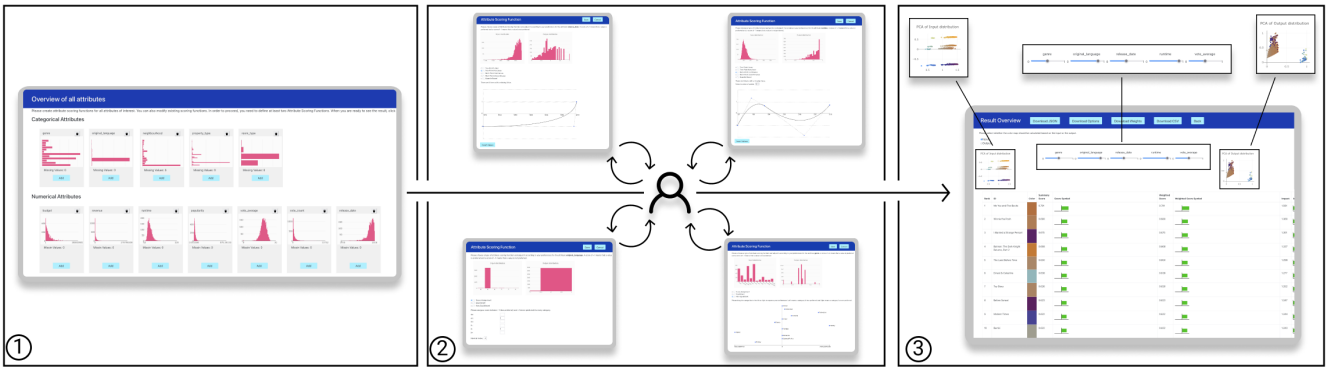


Figure 1: *The workflow used for the creation of a multi-attribute item ranking that is facilitated by RankASco. The left side (1) shows the attribute overview that is at first presented to the user. It facilitates the informed selection of attributes that are used for the creation of the ranking. The center part (2) shows the creation of ASFs for different attributes. The right part (3) shows the weighting and ranking screen where a user can assign weights to the used attributes and inspect the created ranking on the bottom.*

1. RankASco

This supplemental material contains technical implementation details of RankASco as well as an introduction to all visual interactive interfaces. RankASco is publicly available on the website of the Interactive Visual Data Analysis Group of the University of Zurich [†]. For all interfaces that exist in RankASco, screenshots are shown that present the look and working principles of RankASco. The screenshots are created by using the Airbnb dataset which contains a large number of accommodations in the city of Rome [Cox]. We re-use the usage scenario from the main paper for the creation of the screenshots and the illustration of the tool. A second usage scenario is presented in a separate video with the TMDB movie dataset [Dat].

2. Technical Implementation Details

RankASco is a standalone web application that is implemented in JavaScript and with React.js [‡]. The data is stored in local CSV files, therefore no database or back-end is used. The following third-party libraries are used for additional functionality:

For the general styling of RankASco, bootstrap is used [ASS*]. For all histograms and bar charts, the Plotly library [Plo21] is used. The ASF creation interfaces are implemented as plain SVG graphics. For the ranking table, React Table [Lin20] is used. The dimensionality reduction (PCA) is calculated with DruidJS [Cut20]. The data files are parsed with Papa Parse [Hol19]. All libraries are installed through npm [§].

The 2D color map is implemented based on the work of

† https://www.ifi.uzh.ch/en/ivda/research/ranking.html

‡ https://reactjs.org/

§ https://www.npmjs.com/

Dominik Jäckle [JÏ7]. Joshua Comeau [Com20] implemented the basics of the Bézier curves for JavaScript that are used and adapted for this work.

3. Visual Interactive Interfaces

This section covers all visual interactive interfaces of RankASco by showing screenshots of all interfaces and explaining their working principle. The order of the presented interfaces follows the requirements described in the main paper.

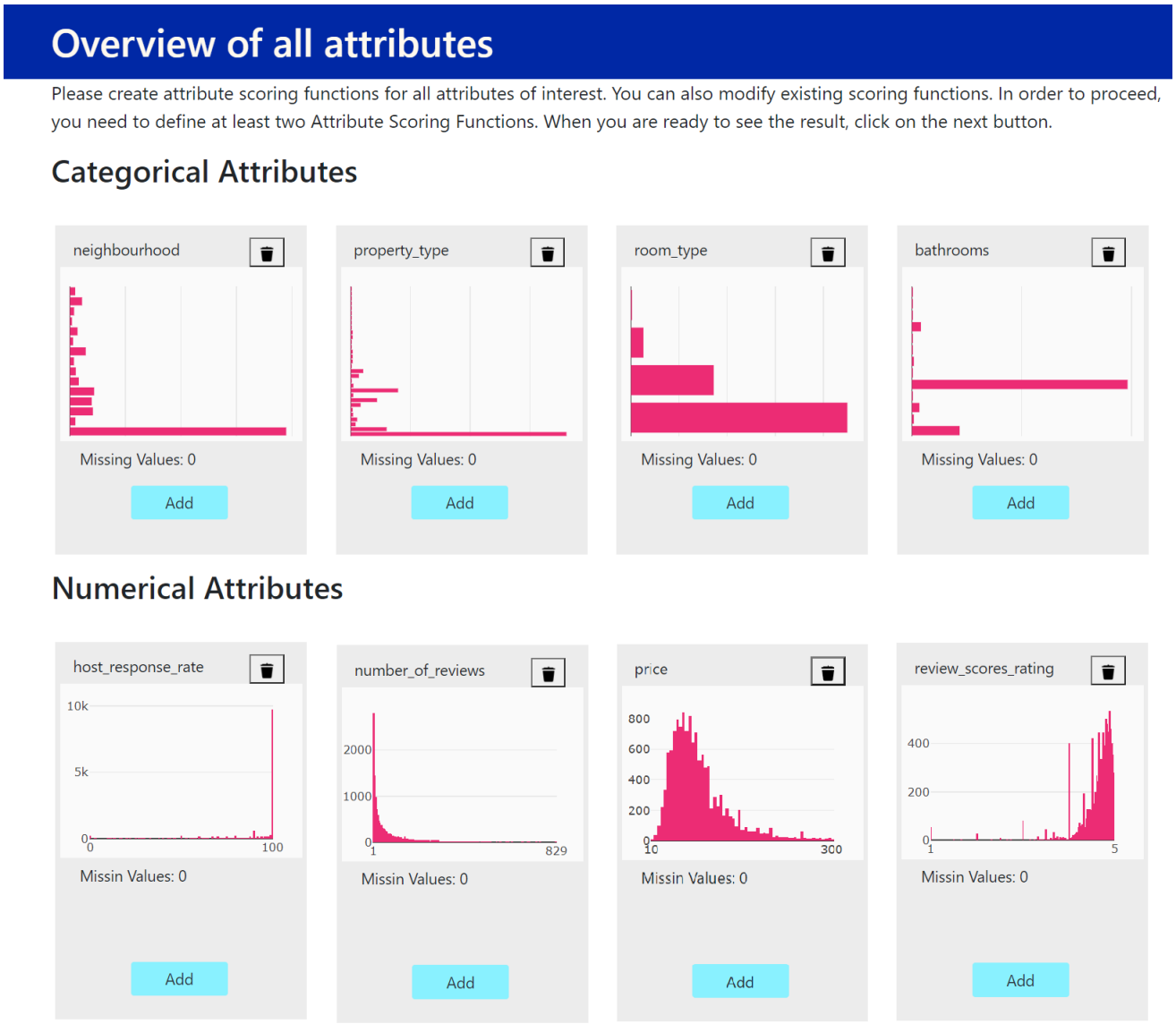


Figure 2: *The overview of all categorical and numerical attributes in the dataset. For every categorical attribute, a bar chart exists that shows the existing categories for the attribute as well as the count for every category. For every numerical attribute, a histogram exists that shows the distribution of the attribute values. By clicking on the* Add *button, users can go to the Attribute Scoring Function Interface for example presented in Figure 4 (categorical) or Figure 7 (numerical) where they can create a new ASF. By clicking on the* Bin *icon, a user can remove an attribute from the dataset.*

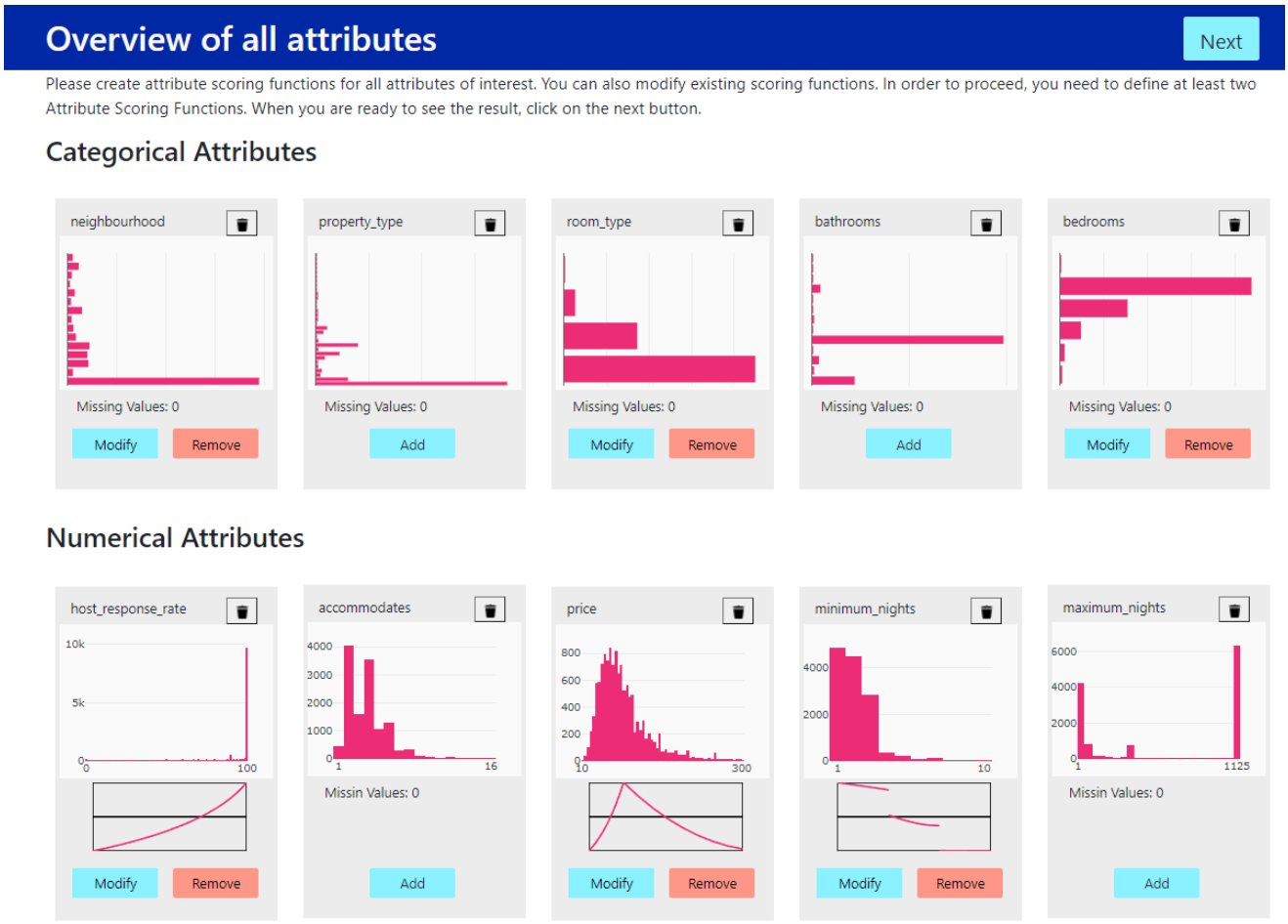


Figure 3: *The attribute overview after the creation of multiple ASFs. For categorical attributes where an ASF has been designed, the* Modify *and* Remove *buttons are shown. By clicking on* Modify*, users can adjust the created ASF. A click on the* Remove *button deletes the ASF. For numerical attributes with an existing ASF, the* Modify *and* Remove *buttons are shown as well as a miniature version of the created ASF. When users have created enough ASFs, they can proceed to the weighting screen by clicking on the* Next *button.*

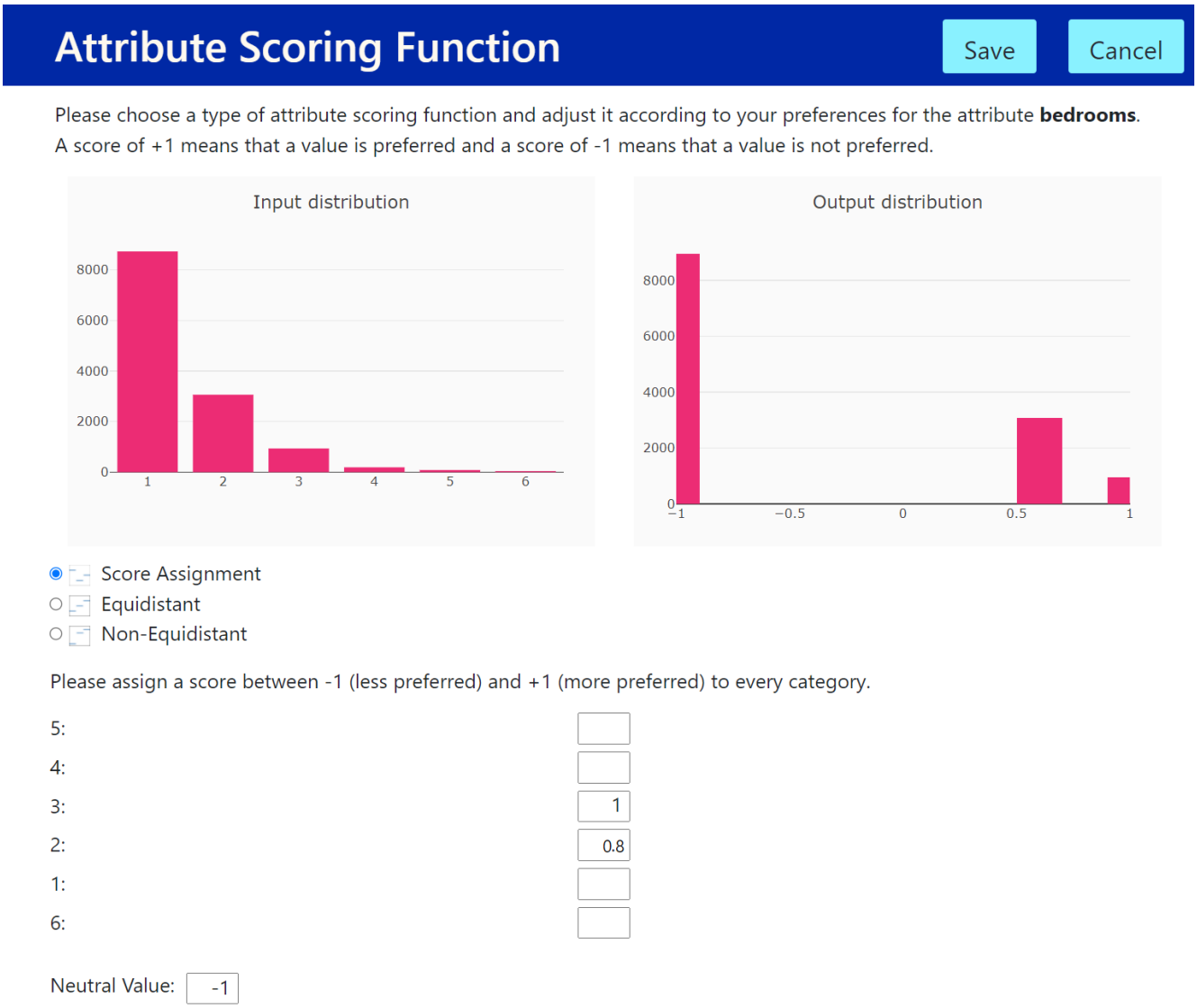


Figure 4: *The score assignment ASF for the attribute bedrooms. On the top left, the input distribution of attribute values is shown. The plot on the top right shows the distribution of output scores which changes in real-time when adjustments are made to the ASF. In the middle, the ASF type selection is shown with radio buttons. On the bottom, the actual ASF interface is shown. This type of ASF consists of a numerical input field for each category where users can enter scores between -1 and +1. For all categories that do not have a score assigned, the neutral value is used which can be set in an additional input field. In this example, 1 was assigned to accommodations that have 3 rooms, 0.8 was assigned to accommodations with 2 rooms, and -1 was assigned to all other accommodations. This could for example be the ASF of a family of four that wants to go on a holiday trip and prefers if the accommodation has one bedroom for the parents and one bedroom for each of the children.*

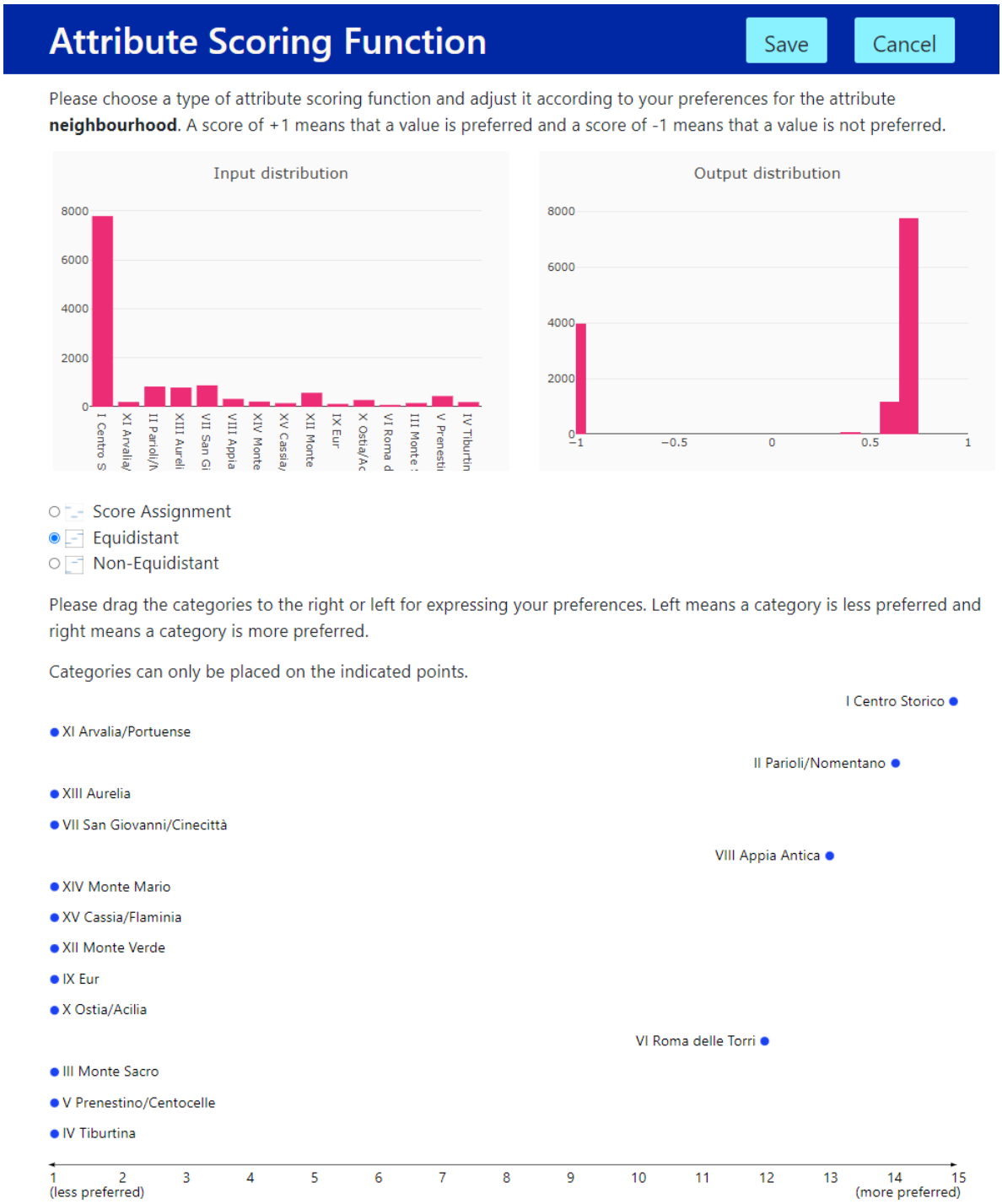


Figure 5: *The equidistant ASF for the attribute neighborhood. On the top left, the input distribution of attribute values is shown. The plot on the top right shows the distribution of output scores which changes in real-time when adjustments are made to the ASF. In the middle, the ASF type selection is shown with radio buttons. On the bottom, the actual ASF interface is shown. This type of ASF shows a coordinate system where all categories are stacked on top of each other in the y-axis direction and the x-axis represents the preference of categories. Users can drag categories along the x-axis where dragging something to the left means, that it is less preferred and dragging something to the right means something is more preferred. Categories can only be dragged onto predefined positions to guarantee that the distances between them are equally sized. It is also possible to only drag some of the categories and leave the others at the default positions. The shown ASF could for example represent the preferences of users that want to stay in the center of Rome for their trip. Therefore, all districts from the center of Rome are dragged to the right while districts that lie further away from the center are kept on the left side of the ASF.*

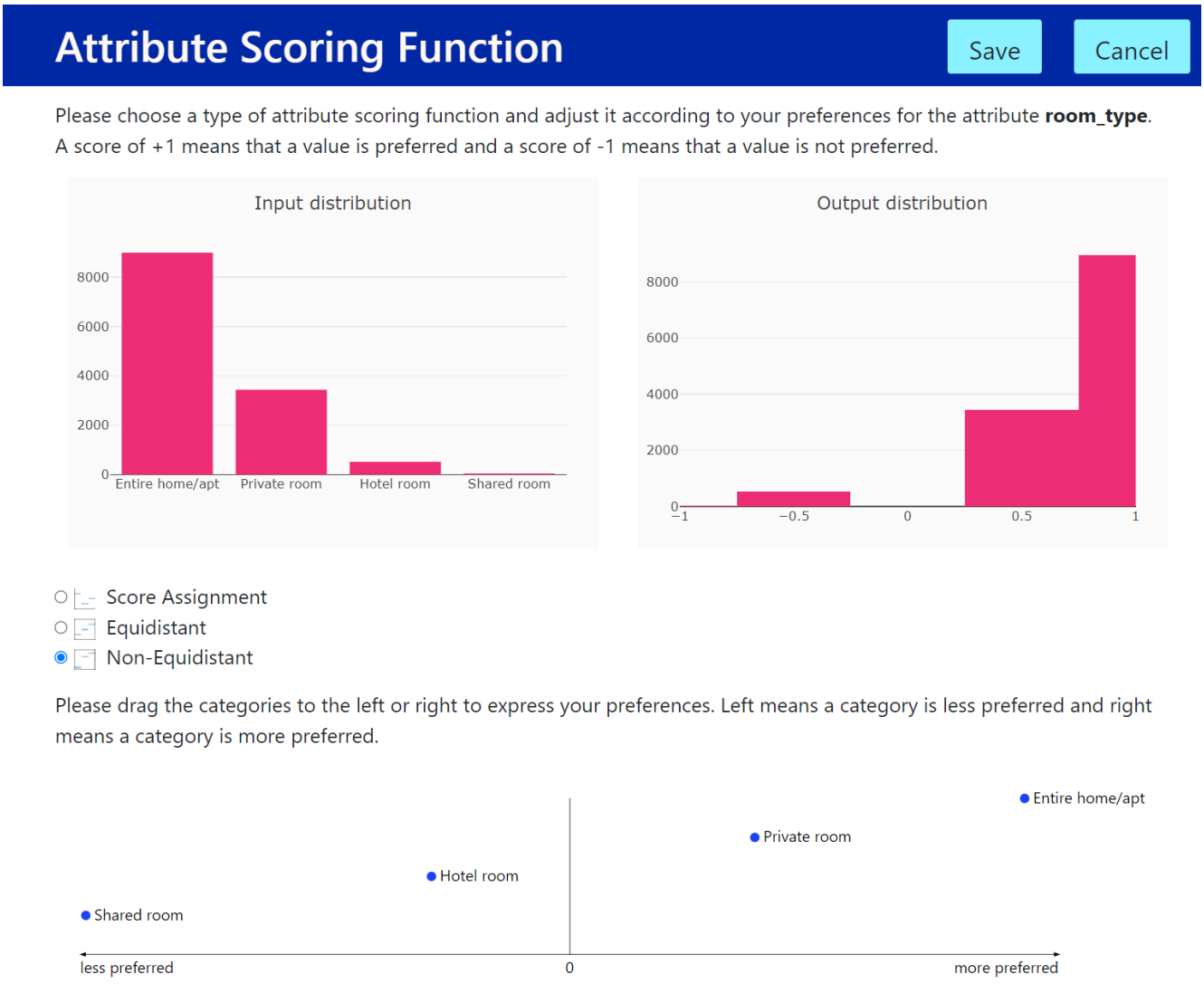


Figure 6: *The non-equidistant ASF for the attribute room_type. On the top left, the input distribution of attribute values is shown. The plot on the top right shows the distribution of output scores which changes in real-time when changes are made to the ASF. In the middle, the ASF type selection is shown with radio buttons. On the bottom, the actual ASF interface is shown. This type of ASF shows a coordinate system where all categories are stacked on top of each other in the y-axis direction and the x-axis represents the preference of categories. Users can drag categories along the x-axis where dragging something to the left means, that it is less preferred and dragging something to the right means something is more preferred. Categories can be dragged freely on the x-axis since the distances between categories do not have to be equally sized. It is also possible, to only drag some of the categories and leave the others at the default positions. The shown ASF could for example represent the preferences for different accommodation types where users prefer to stay either at an entire home or apartment or in a private room since they do not want to share their accommodation with strangers.*

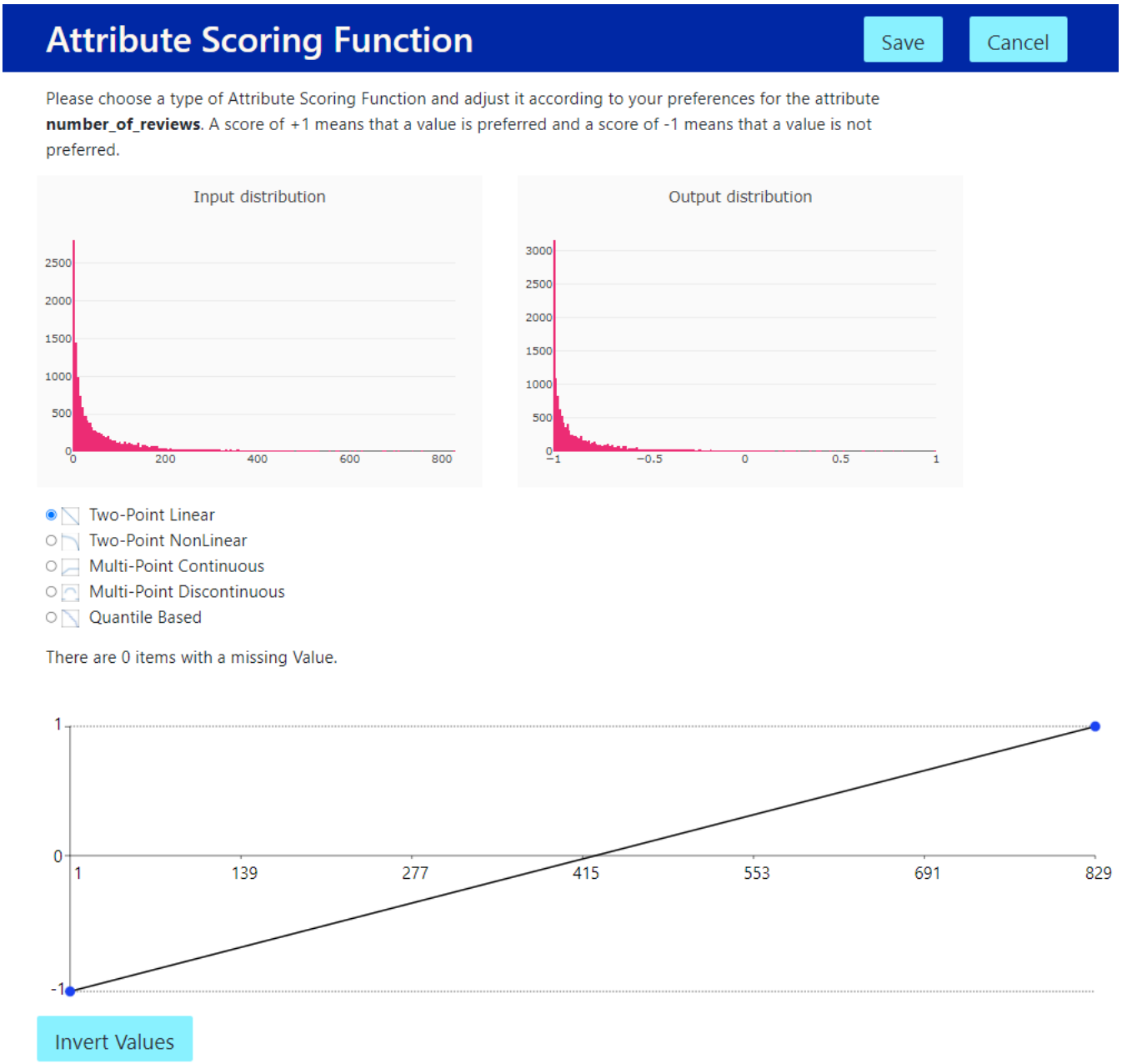


Figure 7: *The two-point linear ASF for the attribute number_of_reviews. On the top left, the input distribution of attribute values is shown. The plot on the top right shows the distribution of output scores which changes in real-time when changes are made to the ASF. In the middle, the ASF type selection is shown with radio buttons. On the bottom, the actual ASF interface is shown. This type of ASF consists of a coordinate system that shows the attribute values on the x-axis and the scores on the y-axis. It contains a linear line that is defined through two points. The line defines the transformation of input values to scores and it can be read like a mathematical function. It can be adjusted by dragging the start or the end point in the vertical direction. The shown ASF could be an example of a user that has a linear preference for accommodations with a high number of reviews.*

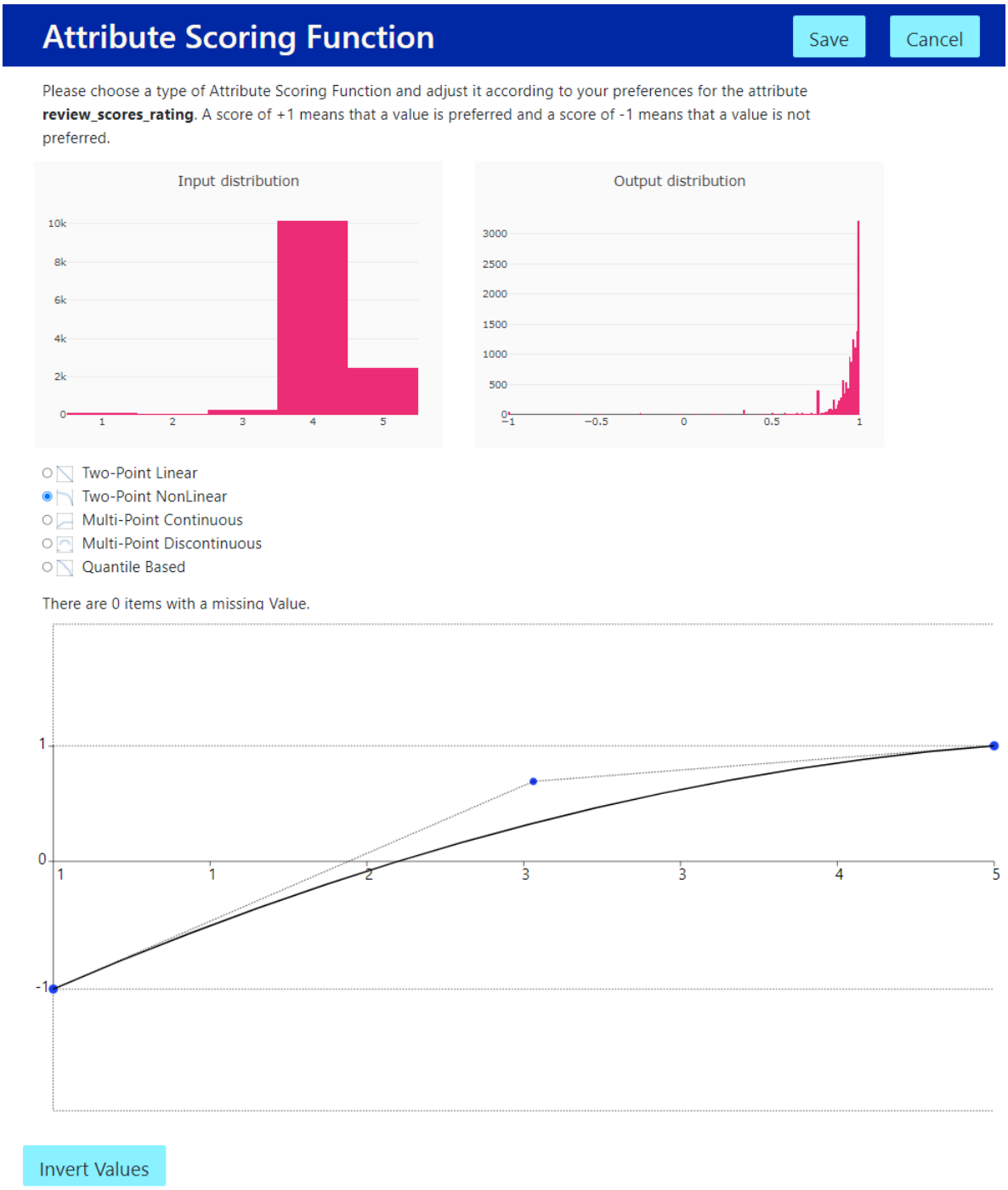


Figure 8: *The two-point non-linear ASF for the attribute review_score_ratings. On the top left, the input distribution of attribute values is shown. The plot on the top right shows the distribution of output scores which changes in real-time when changes are made to the ASF. In the middle, the ASF type selection is shown with radio buttons. On the bottom, the actual ASF interface is shown. This type of ASF consists of a coordinate system that shows the attribute values on the x-axis and the scores on the y-axis. It contains a non-linear line that is defined through two points and one control point that defines the curvature of the line segment. The line defines the transformation of input values to scores based on the working principle of Bézier curves [Won03] and it can be read like a mathematical function. It can be adjusted by dragging the start or the end point into a vertical direction or by dragging the control point either into a vertical or horizontal direction. The shown ASF could be an example of a user that has a non-linear preference for accommodations with a high score where a small change in score makes a disproportional change for the user.*

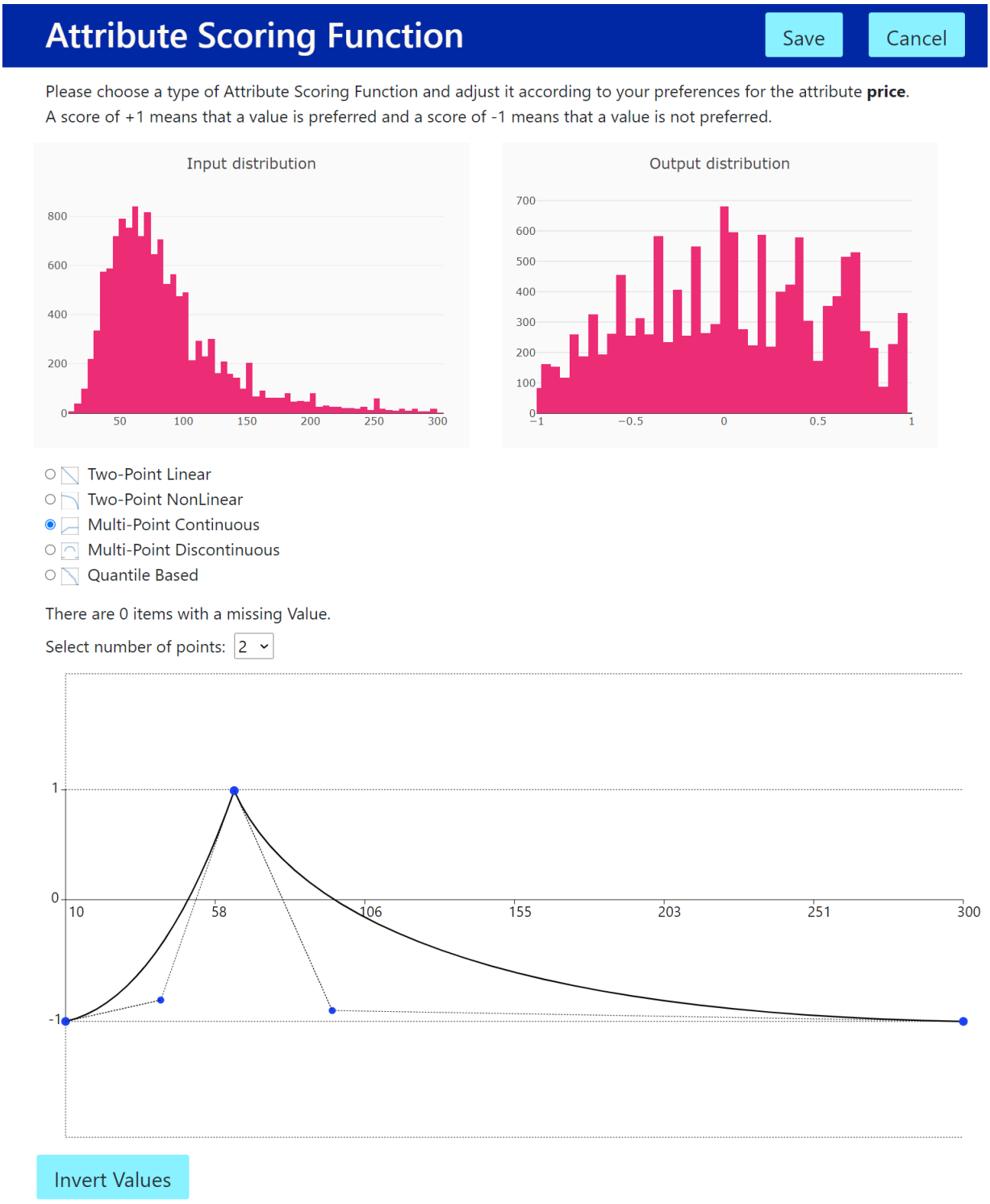


Figure 9: *The multi-point continuous ASF for the attribute price. On the top left, the input distribution of attribute values is shown. The plot on the top right shows the distribution of output scores which changes in real-time when changes are made to the ASF. In the middle, the ASF type selection is shown with radio buttons. On the bottom, the actual ASF interface is shown. This type of ASF consists of a coordinate system that shows the attribute values on the x-axis and the scores on the y-axis. It contains multiple non-linear line segments that are defined through two points and one control point each. The start and end point can be dragged only in a vertical direction while all other points can be dragged in vertical and horizontal directions. Users can change the number of points that the function has through a drop-down menu. The ASF defines the transformation of input values to scores based on the working principle of Bézier curves [Won03] and it can be read like a mathematical function. The shown ASF could be an example for a user that prefers accommodations with a price of around 60€ per night and a price range of 40€ - 90€ that is acceptable for the user.*

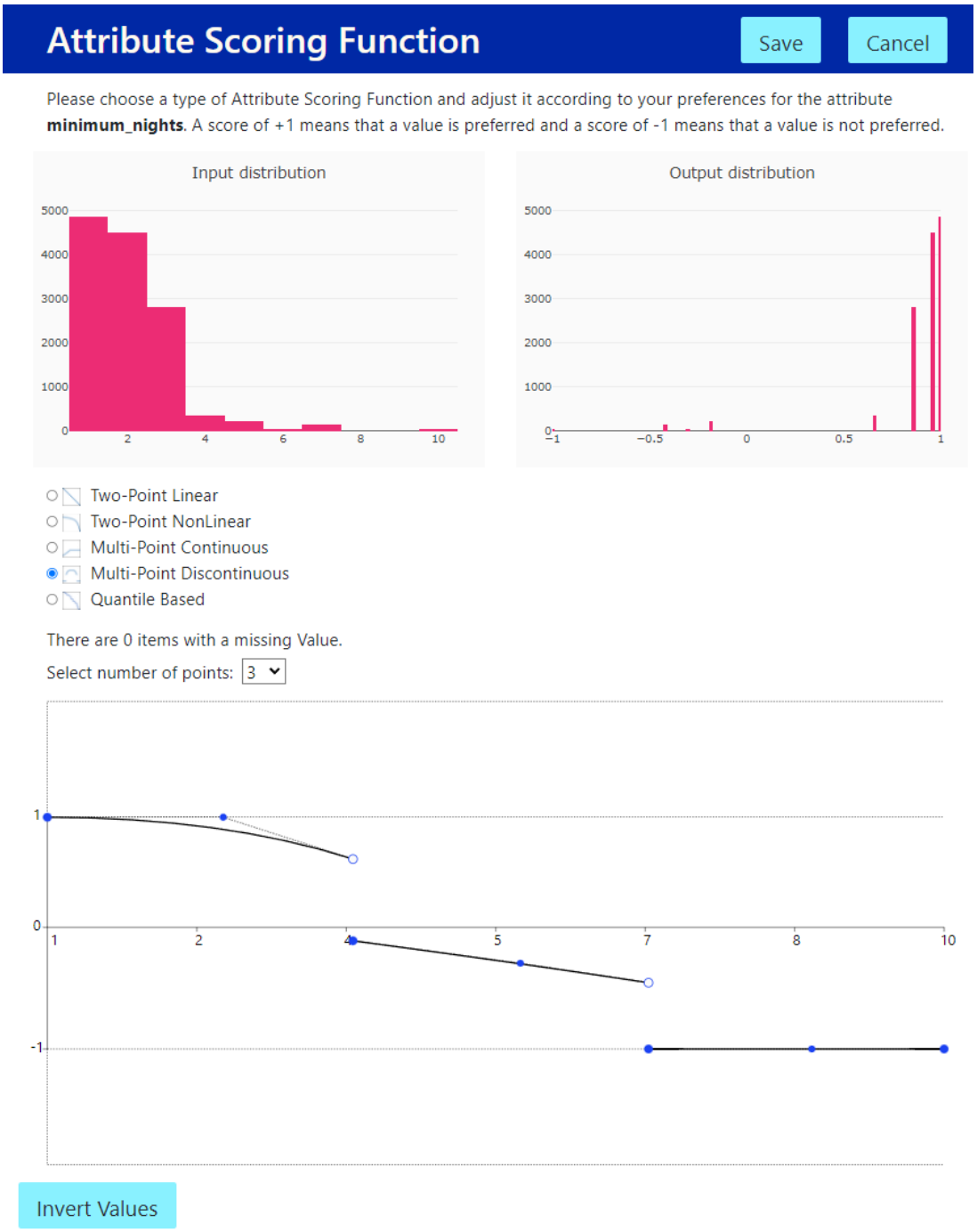


Figure 10: *The multi-point discontinuous ASF for the attribute minimum_nights. On the top left, the input distribution of attribute values is shown. The plot on the top right shows the distribution of output scores which changes in real-time when changes are made to the ASF. In the middle, the ASF type selection is shown with radio buttons. On the bottom, the actual ASF interface is shown. This type of ASF consists of a coordinate system that shows the attribute values on the x-axis and the scores on the y-axis. It contains multiple non-linear line segments that are defined through two points and one control point each. The start and end points can be dragged only in the vertical direction while all other points can be dragged in a vertical and horizontal direction. The individual line segments do not have to be connected in the y-axis direction which allows creating an ASF with discontinuities. Users can change the number of points that the function has through a drop-down menu. The ASF defines the transformation of input values to scores and it can be read like a mathematical function. The shown ASF is an example of the preferences of a user for accommodations that can be booked for a short trip (up to 4 nights) over accommodations that can be booked for a week trip (up to 7 nights) over accommodations that must be booked for a long term.*

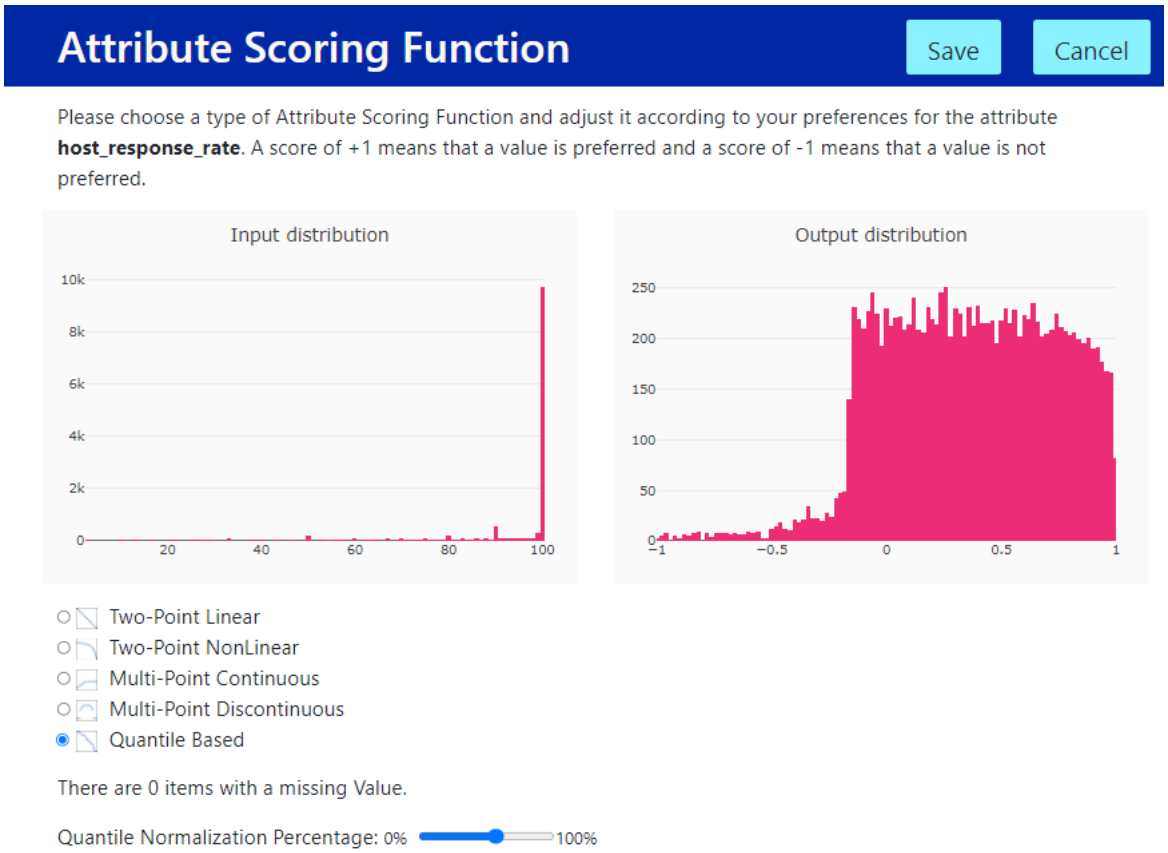
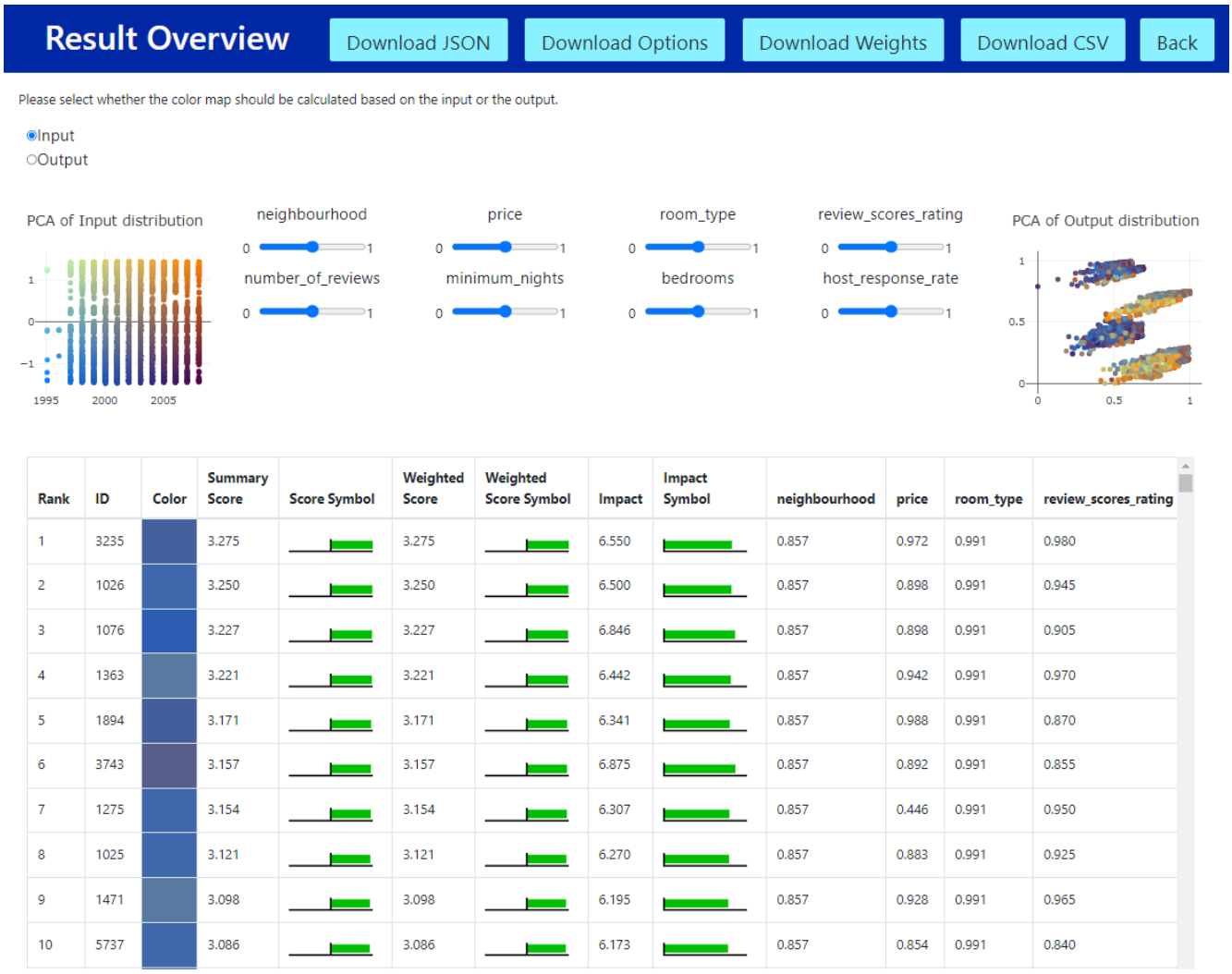
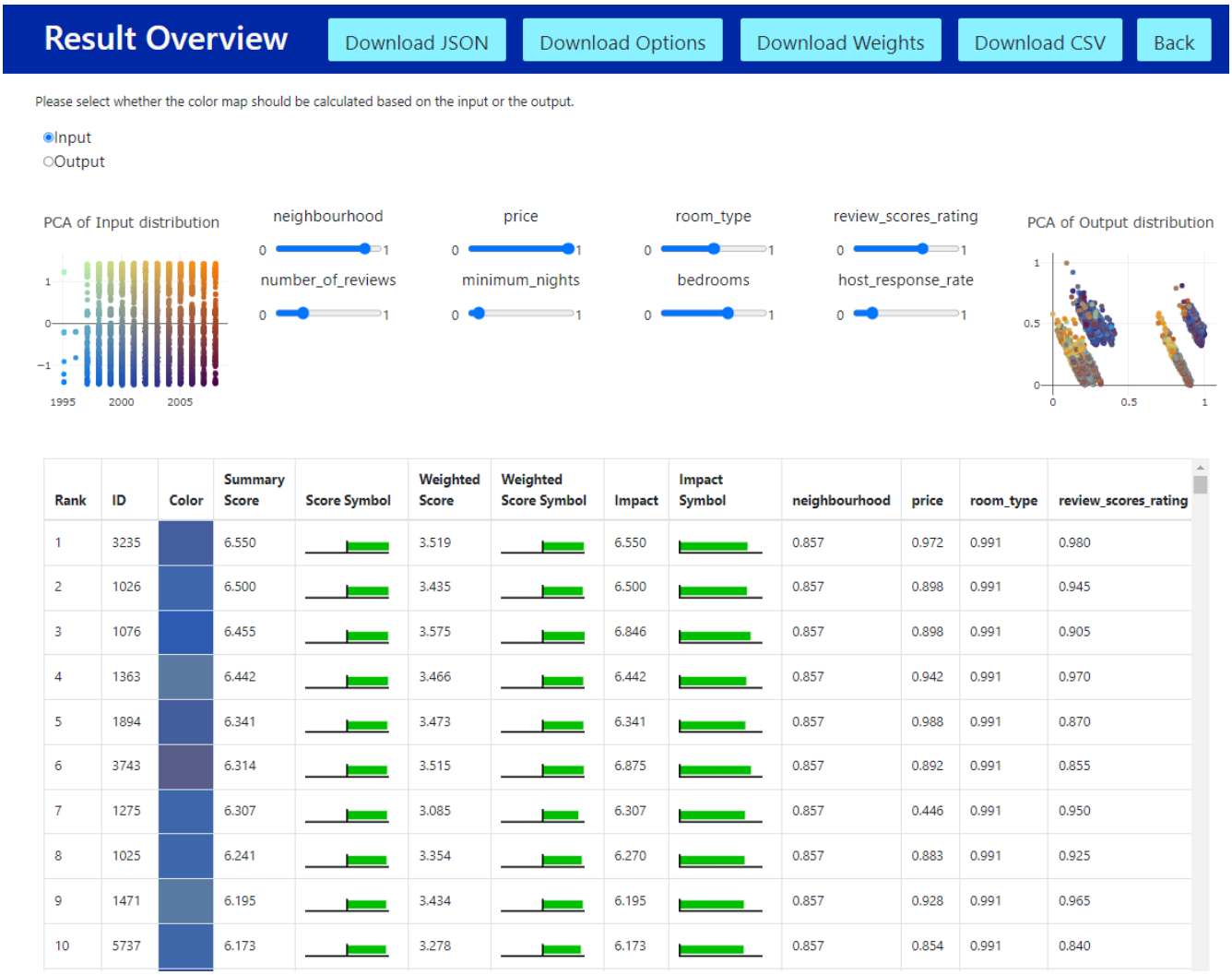


Figure 11: *The quantile-based ASF for the attribute host_response_rate. On the top left, the input distribution of attribute values is shown. The plot on the top right shows the distribution of output scores which changes in real-time when adjustments are made to the ASF. In the middle, the ASF type selection is shown with radio buttons. On the bottom, the actual ASF interface is shown. This type of ASF consists of a slider ranging from 0% to 100% which symbolizes the degree of quantile normalization that is applied to the data. The shown ASF could be an example for users that want to normalize the host response rate attribute since it contains many outliers with small attribute values. By using the quantile-based ASF, these outliers have less weight in the transformation from input values into output scores.*



Rank	ID	Color	Summary Score	Score Symbol	Weighted Score	Weighted Score Symbol	Impact	Impact Symbol	neighbourhood	price	room_type	review_scores_rating
1	3235		3.275		3.275		6.550		0.857	0.972	0.991	0.980
2	1026		3.250		3.250		6.500		0.857	0.898	0.991	0.945
3	1076		3.227		3.227		6.846		0.857	0.898	0.991	0.905
4	1363		3.221		3.221		6.442		0.857	0.942	0.991	0.970
5	1894		3.171		3.171		6.341		0.857	0.988	0.991	0.870
6	3743		3.157		3.157		6.875		0.857	0.892	0.991	0.855
7	1275		3.154		3.154		6.307		0.857	0.446	0.991	0.950
8	1025		3.121		3.121		6.270		0.857	0.883	0.991	0.925
9	1471		3.098		3.098		6.195		0.857	0.928	0.991	0.965
10	5737		3.086		3.086		6.173		0.857	0.854	0.991	0.840

Figure 12: *The weighting and result screen is shown after all ASFs have been created. The header shows the possible download options for the results and created ASFs. The middle part shows the dimensionality reduced input data on the left, the dimensionality reduced output data on the right, and the weight sliders. A color-coding is applied that assigns a color to each item based on a 2D color map [BSM*15]. Users can choose whether the colors should be calculated based on the input or output PCA distribution by a radio button. The weight sliders allow assigning weights between 0 and 1 to each attribute according to its importance to the user. By default, all attributes have a weight of 0,5. The bottom shows the resulting ranking in a tabular view. For each item, its id and color are shown such that it can be identified in the PCA plots. In addition, the different calculated scores are shown in a numerical and symbolical form, and all attribute scores are shown as well in the ranking table.*



Rank	ID	Color	Summary Score	Score Symbol	Weighted Score	Weighted Score Symbol	Impact	Impact Symbol	neighbourhood	price	room_type	review_scores_rating
1	3235		6.550		3.519		6.550		0.857	0.972	0.991	0.980
2	1026		6.500		3.435		6.500		0.857	0.898	0.991	0.945
3	1076		6.455		3.575		6.846		0.857	0.898	0.991	0.905
4	1363		6.442		3.466		6.442		0.857	0.942	0.991	0.970
5	1894		6.341		3.473		6.341		0.857	0.988	0.991	0.870
6	3743		6.314		3.515		6.875		0.857	0.892	0.991	0.855
7	1275		6.307		3.085		6.307		0.857	0.446	0.991	0.950
8	1025		6.241		3.354		6.270		0.857	0.883	0.991	0.925
9	1471		6.195		3.434		6.195		0.857	0.928	0.991	0.965
10	5737		6.173		3.278		6.173		0.857	0.854	0.991	0.840

Figure 13: *The weighting and result screen is shown after all ASFs have been created. In this view, several weights have been adjusted compared to Figure 12 which results in a different PCA for the output scores as well as different scores in the ranking table, and a different order of items in the ranking table.*

References

[ASS*] ABENOJA A., SHEMETOVSKIY A., SIVASUBRAMANYAM A., PASCOE B., TAYMOR C., VIG C., DOZOIS H., O'GRADY M., OIKONOMOU N., IDOL D., MCBRIDE J., QUENSE J., DAGANIO K., HUANG K., TSANG K., M-GOSSELIN M., FUNK M., SMITH M., SCHMITT M., JIA J., KHOMYAKOV N.: React-bootstrap. URL: https://react-bootstrap.github.io/. Last accessed 27 July 2021. 1

[BSM*15] BERNARD J., STEIGER M., MITTELSTÄDT S., THUM S., KEIM D., KOHLHAMMER J.: A survey and task-based quality assessment of static 2d colormaps. vol. 9397, SPIE Press, pp. 93970M–93970M–16. doi:10.1117/12.2079841. 13

[Com20] COMEAU J.: Dynamic bézier curves, 2020. URL: https://www.joshwcomeau.com/animation/dynamic-bezier-curves/. Last accessed 27 July 2021. 2

[Cox] COX M.: Inside airbnb. URL: http://insideairbnb.com/index.html. 1

[Cut20] CUTURA R.: Druidjs, 2020. URL: https://saehm.github.io/DruidJS/index.html. Last accessed 27 July 2021. 1

[Dat] DATABASE T. M.: Tmdb 5000 movie dataset. URL: https://www.kaggle.com/tmdb/tmdb-movie-metadata. 1

[Hol19] HOLT M.: Papa parse, 2013-2019. URL: https://www.papaparse.com/. Last accessed 27 July 2021. 1

[JÏ7] JÄCKLE D.: 2d colormaps javascript plugin, 2017. URL: http://dominikjaeckle.com/projects/color2d/. Last accessed 27 July 2021. 2

[Lin20] LINSLEY T.: React table, 2020. URL: https://react-table.tanstack.com/. Last accessed 27 July 2021. 1

[Plo21] PLOTLY: Plotly, 2021. URL: https://plotly.com/javascript/. Last accessed 27 July 2021. 1

[Won03] WONG B. D.: Bézierkurven: gezeichnet und gerechnet: Ein elementarer zugang und anwendungen. 9, 10