Decision Analysis report 3 - Preference Learning

Marcin Gólski Piotr Kaszubski

May 11, 2023

Contents

1 Dataset **2**
1.1 Overview . . . 2
1.2 Transformation . . . 2
1.3 Sample . . . 2
1.4 Splitting into train and test sets . . . 3
1.5 The "3-alternative" analyses . . . 3
1.6 Extras . . . 3

2 Summary of all models **4**

3 Task 1 – XGBoost **5**
3.1 Model visualization . . . 5
3.2 Preference analysis . . . 5
3.3 3-alternative analysis . . . 6
3.3.1 Analytical approach . . . 6
3.3.2 Space sampling . . . 7
3.3.3 Variable contribution plots . . . 8

4 Task 2 – ANN-UTADIS **8**
4.1 Model visualization . . . 9
4.2 Preference analysis . . . 10
4.3 3-alternative analysis . . . 10
4.3.1 Analytical approach . . . 10
4.3.2 Space sampling . . . 10
4.3.3 Variable contribution plots . . . 11

5 Task 3 – ANN **12**
5.1 Model visualization . . . 12
5.2 Preference analysis . . . 12
5.3 3-alternative analysis . . . 13
5.3.1 Analytical approach . . . 13
5.3.2 Space sampling . . . 13
5.3.3 Variable contribution plots . . . 14

1 Dataset

We chose the **car evaluation** dataset from the *UCI Machine Learning Repository*[1], due to its simplicity and alignment with the project specification.

1.1 Overview

The dataset comprises of **1728 instances**, each having **6 features** and belonging to one of **4 classes**. Each feature is monotonic, with some being categorical, and some numerical.

1.2 Transformation

In order to make the data suitable for the project, it was necessary to transform the features and classes to monotonic numerical values and normalize them. This was done in the following way:

feature	**type**	**values**	
		before	**after**
buying, *maint*	cost	`low`	`0.000000`
		`med`	`0.333333`
		`high`	`0.666667`
		`vhigh`	`1.000000`
doors	gain	`2`	`0.000000`
		`3`	`0.333333`
		`4`	`0.666667`
		`5more`	`1.000000`
persons	gain	`2`	`0.000000`
		`4`	`0.500000`
		`more`	`1.000000`
lug_boot	gain	`small`	`0.000000`
		`med`	`0.500000`
		`big`	`1.000000`
safety	gain	`low`	`0.000000`
		`med`	`0.500000`
		`high`	`1.000000`
class	–	`unacc`	`1.000000`
		`acc`	`2.000000`
		`good`	`3.000000`
		`vgood`	`4.000000`

Note that for some tasks, the decision class will be binarized to represent whether a car is **at least acceptable** (`unacc` → `0.000000`, everything else → `1.000000`).

1.3 Sample

(the first columns enumerate rows and are not part of the data)

1	vhigh	vhigh	2	2	med	med	unacc
2	high	low	5more	2	small	low	unacc
3	med	low	4	4	med	high	vgood
4	low	low	4	more	med	med	good
5	vhigh	vhigh	5more	more	med	med	unacc

Table 2: Raw sample, as present in the original CSV file

[1] https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

#	*buying*	*maint*	*doors*	*persons*	*lug_boot*	*safety*	*class*
1	1.000000	1.000000	0.000000	0.000000	0.500000	0.500000	0.000000
2	0.666667	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
3	0.333333	0.000000	0.500000	0.500000	0.500000	1.000000	4.000000
4	0.000000	0.000000	0.500000	1.000000	0.500000	0.500000	3.000000
5	1.000000	1.000000	1.000000	1.000000	0.500000	0.500000	0.000000

Table 3: Processed sample

The processed dataset is available at `data/car-evaluation.csv` (from the project root directory).

1.4 Splitting into train and test sets

In order to be able to compare all models fairly, it is essential to guarantee that their respective training and testing sets are identical. This is achieved with SciKit-Learn's `train_test_split()` function, by fixing its random seed. `train_test_split()` also takes care of the stratification of the resulting sets.

1.5 The "3-alternative" analyses

For all models' "3-alternative" analyses we used the same 3 alternatives (again, to retain fairness of comparison):

#	*buying*	*maint*	*doors*	*persons*	*lug_boot*	*safety*	*class*
1	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000
2	0.000000	0.000000	0.000000	0.500000	1.000000	0.000000	1.000000
3	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	4.000000

Table 4: Alternatives chosen for all 3-alternative analyses

As you can see, the first alternative represents the worst case, the third the best case, and the second one lies somewhere in the middle.

1.6 Extras

Throughout the making of this report, we've accumulated some plots that we did not think would contribute much in terms of quality, and would take up many extra pages. Most notably this happened with the ceteris paribus plots of the 3-alternative analyses.

In such cases, we either completely or partially omit a plot from the report. If you are interested in those omitted parts, take a look at the `report/img/extras/` (from project root directory).

2 Summary of all models

Before focusing on all tasks individually, here are some metrics of our final results.

Model	Accuracy	AUC	F1
XGBoost	0.9653	0.9908	0.9434
ANN-UTA	0.7803	0.8669	0.6545
ANN	0.8815	0.8391	0.7960

Table 5: Performance of various models during testing

Over the course of this report three basic types of visualizations provided by the DALEX library will be used. Here is a brief explanation:

- Feature importance

  DALEX employs permutation-based variable-importance. The explainer shuffles the values of selected attribute and computes the difference between the performance of the model trained on this altered data and the original: $L_{\text{perm}} - L_{\text{org}}$. The greater the difference, the more important the feature is. AUC loss function is used in this context.

- Ceteris paribus

  They demonstrate how changing the value of one feature while leaving others unchanged would affect the output of a model. The value of the feature is presented on the x-axis, and the model prediction with this setting - on y-axis. The original position is indicated with a dot.

- Variable contribution

  The intercept is the average evaluation of all examples. Going down the chart adds more features to a "filter" that alternatives for which the average model evaluation is computed must pass. The changes in these averages represent how making the alternative's description more specific leads to the output the model produces when the alternative is fed to it.

3 Task 1 – XGBoost

The model was trained using mostly default parameters of the Python `xgboost` module. Despite this, it managed to achieve quite tremendous results, as visible in table 5.

3.1 Model visualization

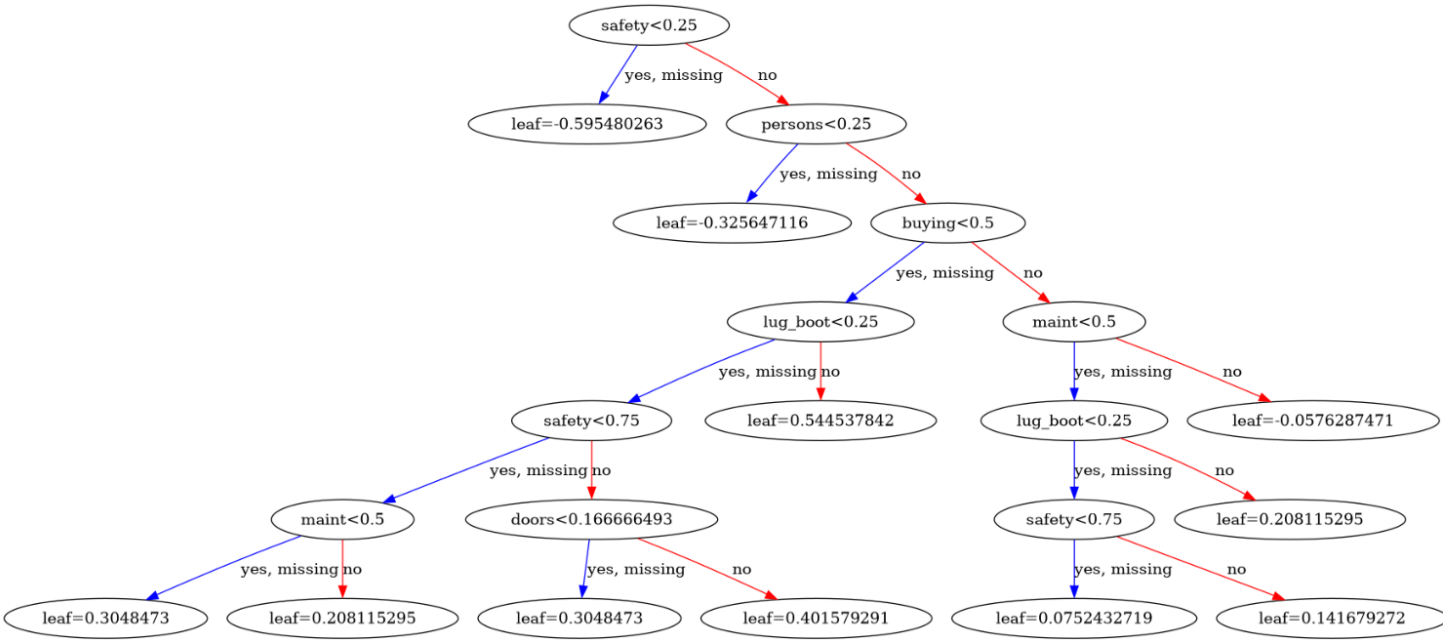


Figure 1: Last tree in the ensemble produced by XGBoost

3.2 Preference analysis

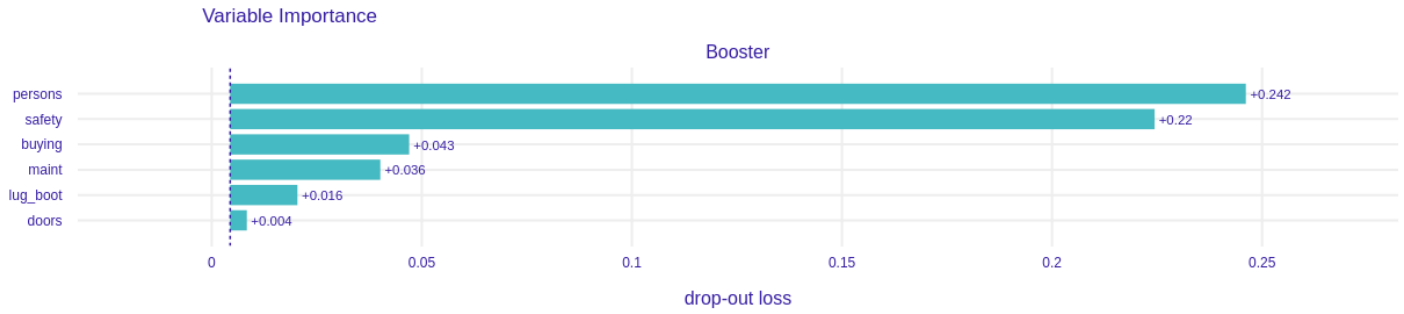


Figure 2: XGBoost feature importance as computed by DALEX

DALEX assigns very high importance to *persons* and *safety*, in that order. This aligns with figure 4 of the forthcoming ceteris paribus analysis. These importances don't fully agree with the above model visualization (in which *safety* seems to play a more significant role), but it is important to keep in mind that the visualization is merely of a single tree in an ensemble.

XGBoost The XGBoost library also has native support for quantifying and plotting feature importance. It must use a slightly different method than DALEX, because the assigned values and even the ranking of the features are not the same.

For the sake of inter-model comparability, we prioritized the DALEX output, but nevertheless this interpretation is also interesting:

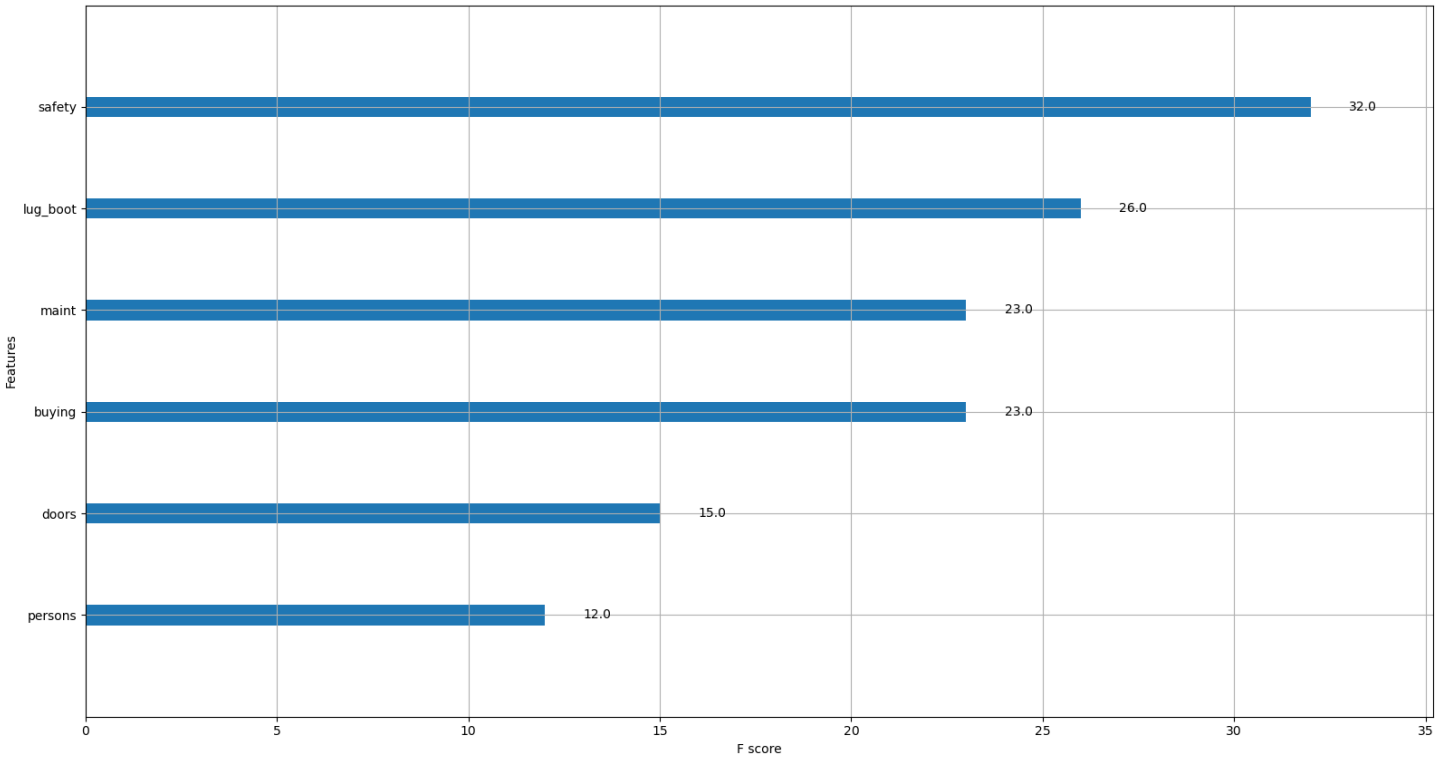


Figure 3: Importance of each feature, according to XGBoost

What is quite frankly shocking is the assignment of the lowest importance score to *persons*. This runs counter to both our intuitive expectations and other analyses.

3.3 3-alternative analysis

3.3.1 Analytical approach

Predicting the outcome of a tree ensemble method is certainly not as trivial as doing so with only a single tree. However, for the sake of simplicity, I will assume that a single tree is a sufficiently accurate approximation of the entire model and make judgements based on the final tree in the ensemble produced by XGBoost, as seen on figure 1.

Alternative 1 This alternative has 0 *safety*, which disqualifies it on the very first tree branch (we want a positive leaf value). Following the model's parameters, it would need a *safety* of at least 0.25. Using the same logic for subsequent branches, *persons* would need to be at least 0.25 as well. From there the tree becomes more complex, so I will assume a simple greedy strategy and only tweak feature values when absolutely necessary. We proceed to the right with *buying* unchanged. Then we must lower *maint* by more than 0.5. After that, all leaves are positive, so there is no need for further changes.
To sum up:

- *safety* 0.0 $\rightarrow$ 0.25
- *persons* 0.0 $\rightarrow$ 0.25
- *maint* 1.0 $\rightarrow$ 0.49

Alternative 2 Using the same procedure as with alternative 1, we get:

- *safety* 0.0 $\rightarrow$ 0.25

Alternative 3 Same procedure once more, but this time we are aiming for the unacceptable class:

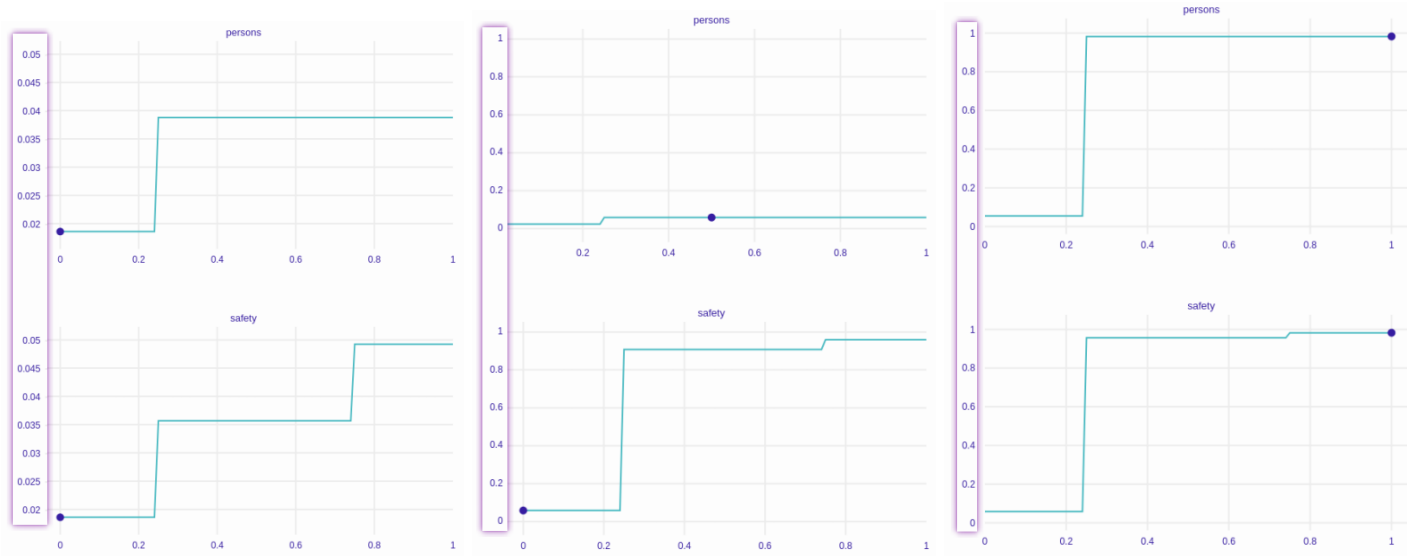
- *safety* 1.0 → 0.24

3.3.2 Space sampling

To concisely show how changing the value of just one attribute would change the result, ceteris paribus analysis was performed.

Most plots were approximately "flat", with no major shifts in prediction (Y-axis). What this generally means is that those features alone can't swivel the classification of an alternative.

There are, however, 2 features that do exhibit some major shifts – *safety* and *persons*. Files with all plots are available in `report/img/extras`. In order not to overcrowd the report, only the plots for *safety* and *persons* for the 3 alternatives are presented.



(a) alternative 1 (worst) (b) alternative 2 (mid) (c) alternative 3 (best)

Figure 4: Ceteris paribus of *persons* and *safety* features in XGBoost

For alternative 2, *safety* is very highly correlated with the decision class. There is a clear boundary at around 0.25 where the model leaps from being barely above 0 to being almost 1. This suggests that there were many middle alternatives which were unacceptable solely due to a low *safety* value. The same observation can be made for alternative 3 (both *persons* and *safety*).

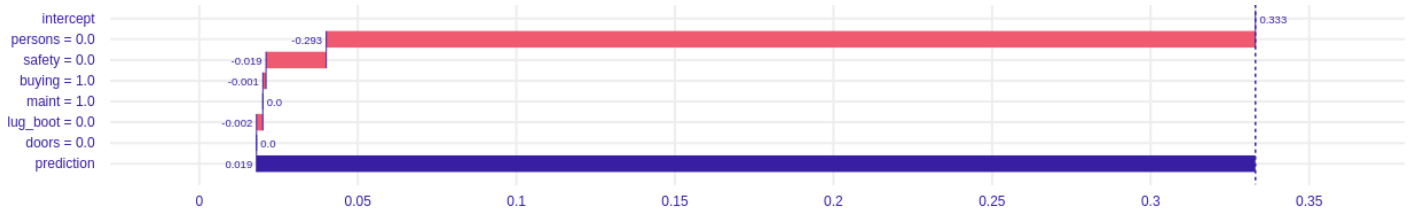
A bonus observation is that these plots resemble **step functions**. That is because XGBoost is a tree ensemble model, and trees are unable to produce smooth decision boundaries due to their nature of branching on specific feature values.

Alternative 1 Applying the changes suggested in the *Analytical approach* section did not produce the wanted change, but it did increase the overall score from `0.02` to `0.38` (on a range from `0.0` to `1.0` and a binary decision boundary `0.5`).

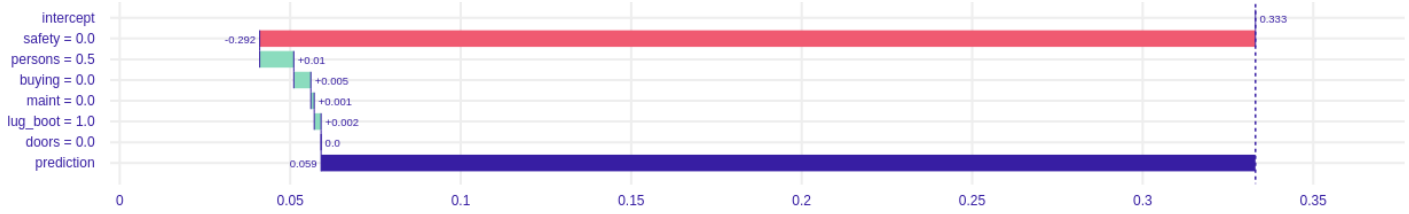
Alternative 2 Increasing *safety* to 0.25 yielded a change from `0.06` to a whopping `0.91`. This further reinforces the idea that *safety* plays a dominant role in predicting the outcome class.

Alternative 3 Decreasing *safety* to 0.24 reduced the score from `0.98` to `0.06`. Once again, *safety* proves critical.

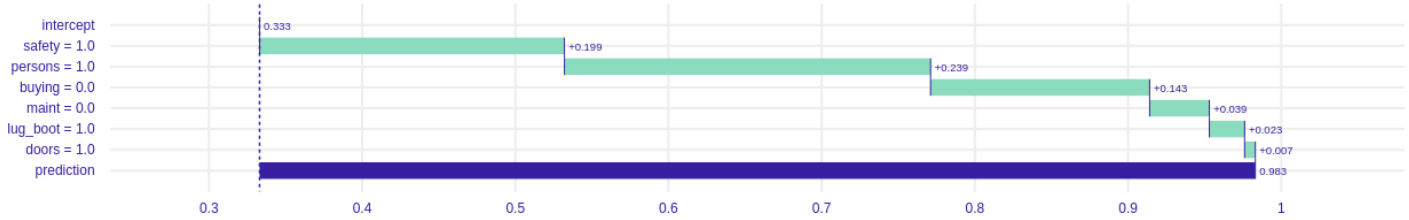
3.3.3 Variable contribution plots



(a) alternative 1 (worst)



(b) alternative 2 (mid)



(c) alternative 3 (best)

Figure 5: Variable contributions of 3 alternatives in XGBoost

The variable contribution plots reveal a striking imbalance between the features in the XGBoost model. For low and mid ranking alternatives, almost all of the negative feedback is attributed to *safety* or *persons*, with remaining features hardly affecting the overall prediction score.

The high ranking alternative is much more uniform, with *doors* having the smallest contribution.

4 Task 2 – ANN-UTADIS

The trained model heavily utilised code provided in `ANN-UTADIS.ipynb` and `helpers.py`. Small changes were introduced to suit the authors' preferences better, but the overall architecture is based entirely on these snippets. The model analysed in this section contains 12 hidden layers.

It is essential to note here that the two cost type criteria: *buying* and *maint* have been transformed in the following way: `new_value = 1 - old_value`. Thanks to this procedure more preferred values are represented by larger numbers, just as in the case of gain type criteria. This is done in order to facilitate using non-decreasing marginal value functions for all criteria.

4.1 Model visualization

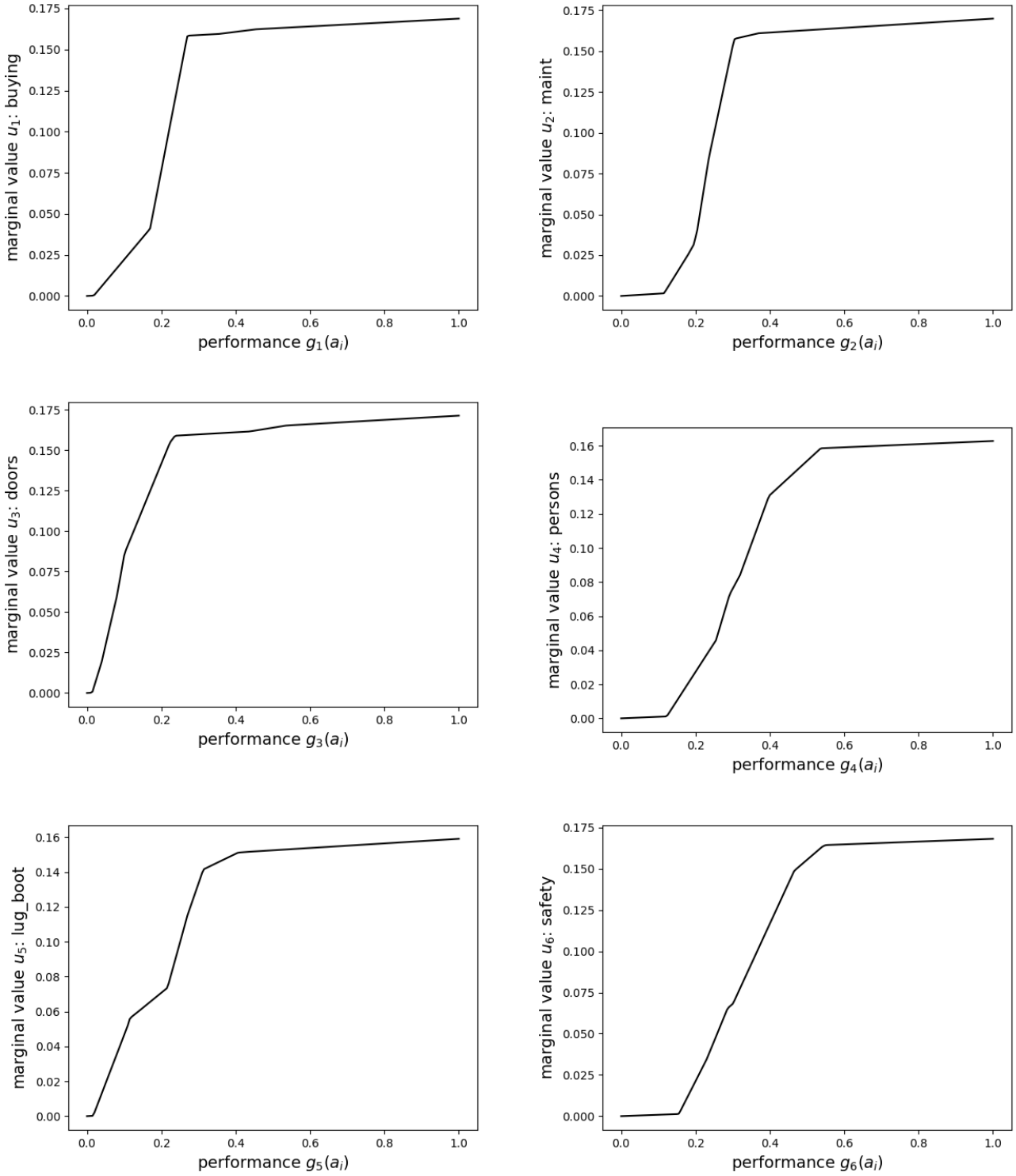


Figure 6: Marginal value functions for all criteria generated by the network

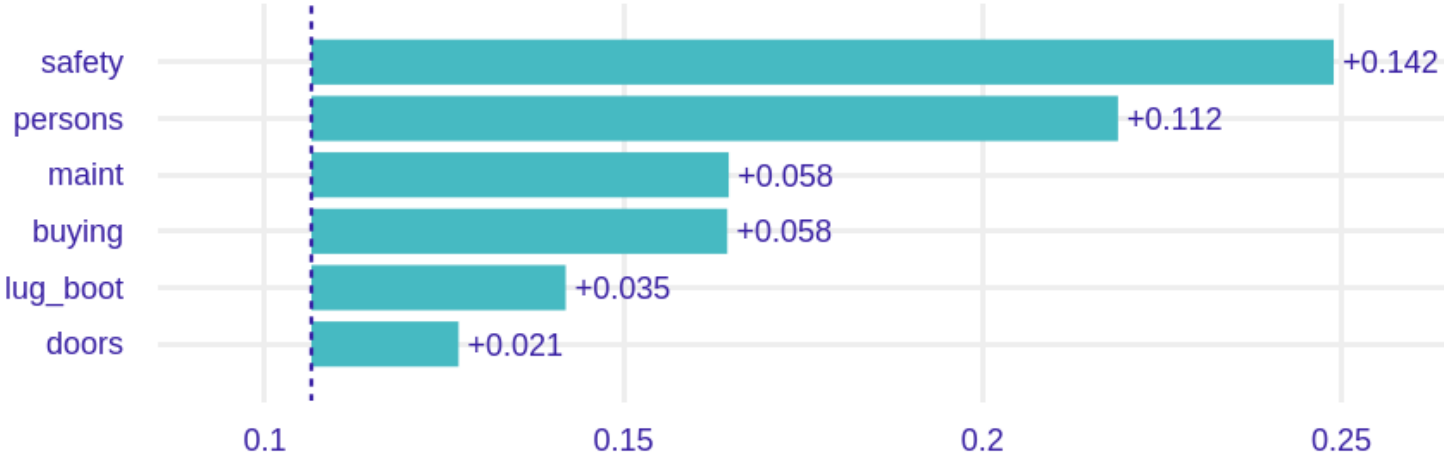


Figure 7: ANN-UTADIS feature importance as computed by DALEX

4.2 Preference analysis

The UTA model has the advantage that it utilises marginal preference functions, which can be very directly interpreted as an expression of DM's preference. Looking at them, no criterion seems to have a decisive influence or be irrelevant. The maximum values of their marginal value functions ("weights") are very similar - they are all concentrated roughly between 0.16 and 0.18.

All but one functions are more or less flat for larger values of g_i, suggesting that the decision maker is rather indifferent regarding whether an alternative is exceptional on one criterion. In other words, the DM is interested in balanced alternatives. This claim is reinforced by the results of ceteris paribus analysis - changing the value of just one parameter never has a significant effect on the model's overall prediction.

An especially interesting marginal value function is the one regarding criterion u_2 *doors*. Having 3 doors seems to be a very big advantage over having only 2, but adding more doesn't give much benefit, as seen on its plot. This is also reflected in the ceteris paribus plots discussed later.

The feature importance plot generated by DALEX is significantly less uniform. *doors* seems yet again to be the most interesting attribute, with its low importance. I think that this might be because a large range of performance values result in almost no change in the value of u_i.

4.3 3-alternative analysis

4.3.1 Analytical approach

We can use the marginal value functions to get an insight into how the model performs classification. The model would assign an object to a different class if their sum (comprehensive value) is smaller than 0.5. Based on that we can make the following inferences: - *best* can't be classified as unacceptable by changing the value on just one criterion - *worst* can't be classified as acceptable by changing the value on just one criterion

For *mid*, things are a bit more complicated. As is, it is assigned 0 (unacceptable). However, it attains the maximum possible score for *buying*, *maint*, and *lug_boot*. It's 0.5 on *persons* should already give it most of the "score" from this criterion, so improvement should be sought elsewhere. We are left with *doors* and *safety*, which are originally at 0 for *mid*. I suspect, that changing *doors* to 0.25 could be enough. It should certainly do more than adding the same value to *safety*, due to the sharper increase of the marginal value function of the former. Running the model for such two alternatives confirms this assessment. With *doors* = 0.25, the model classifies the alternative as barely acceptable with 0.0009, while changing *safety* to 0.25 maintains negative judgement with -0.1128.

4.3.2 Space sampling

To concisely show how changing the value of just one attribute would change the result, ceteris paribus analysis was performed.

Files with all plots are available in `report/img/extras`. In order not to overcrowd the report, only the plots for *doors* for the 3 alternatives are presented.

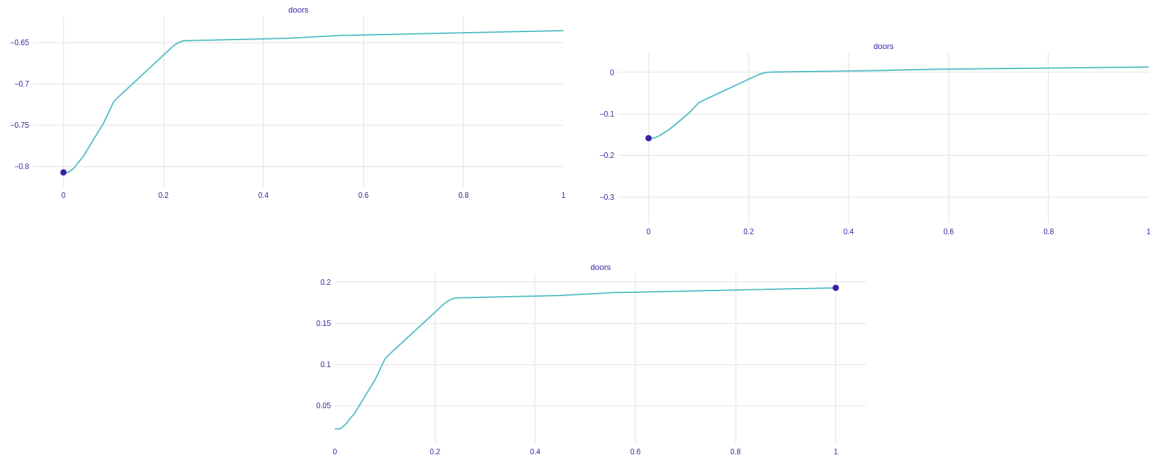
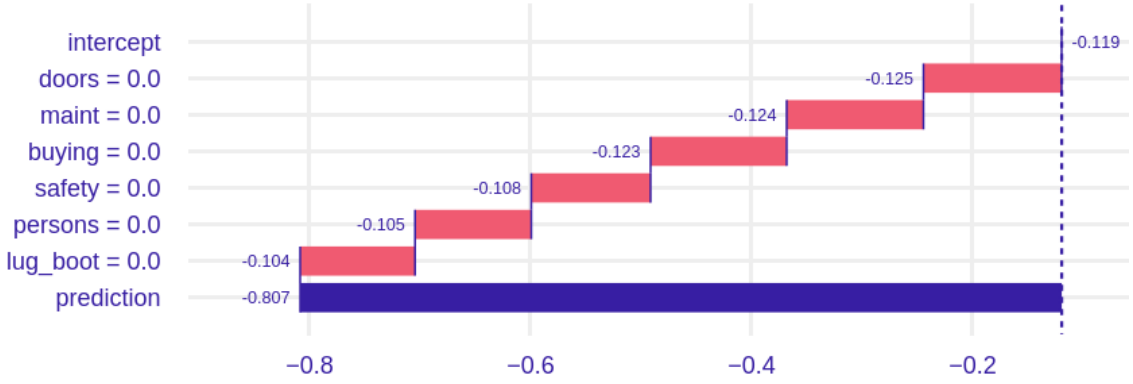


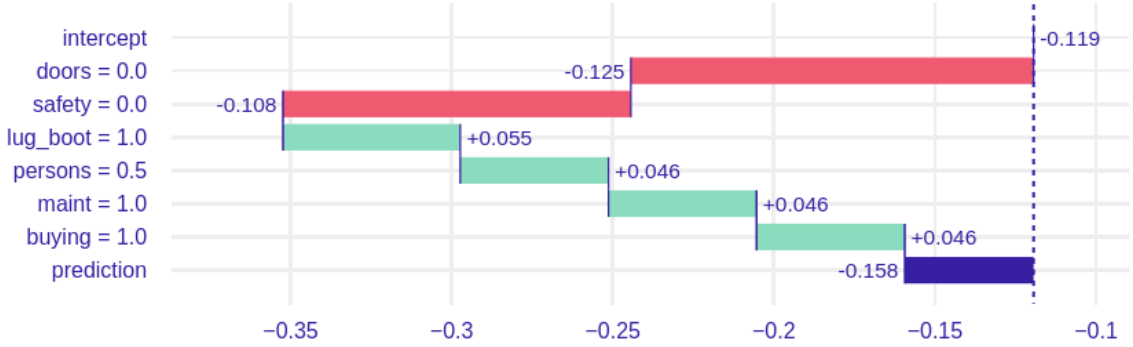
Figure 8: Ceteris paribus analysis for criterion *doors* for (from left to right and top to bottom)

In image form, the plots can't be examined as thoroughly as in interactive DALEX output, but it agreed with the earlier theoretical prediction, and measured model output. It showed that *doors*=0.25 would result in *mid* being classified as acceptable. It also allows us to see, that with *doors*=0.23, the model is expected to produce -0.001, but that changing the value of this attribute to 0.24 might be sufficient to obtain a positive result.

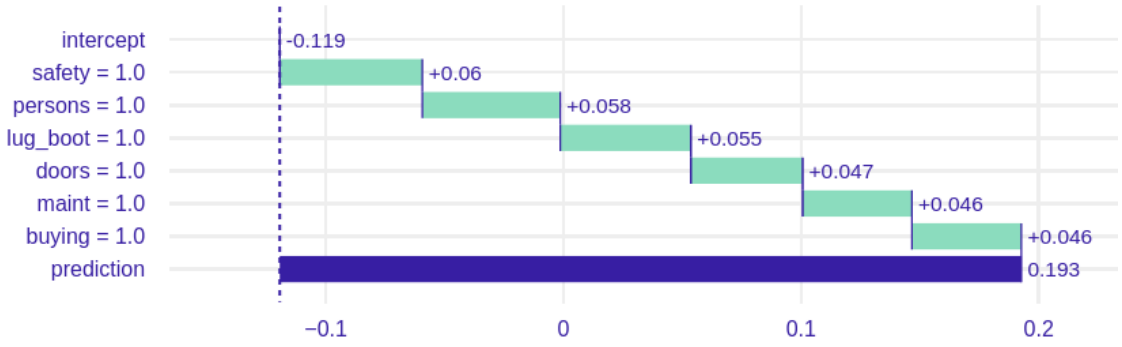
4.3.3 Variable contribution plots



(a) Variable contribution of alternative 1 (worst) in ANN-UTADIS



(b) Variable contribution of alternative 2 (mid) in ANN-UTADIS



(c) Variable contribution of alternative 3 (best) in ANN-UTADIS

5 Task 3 – ANN

The model used here is a basic Artificial Neural Network with 12 dense layers, using LeakyReLU as activation function. The output layer applies the sigmoid function, with 1 representing full confidence in assigning the object to class 1, and 0 - full confidence in assigning it to 0.

Importantly for the plots seen later, this model also used the $1 - \text{old\_value}$ trick for cost criteria. This was done to make the model a more comparable to ANN-UTADIS (for the same reason, it has the same number of layers).

5.1 Model visualization

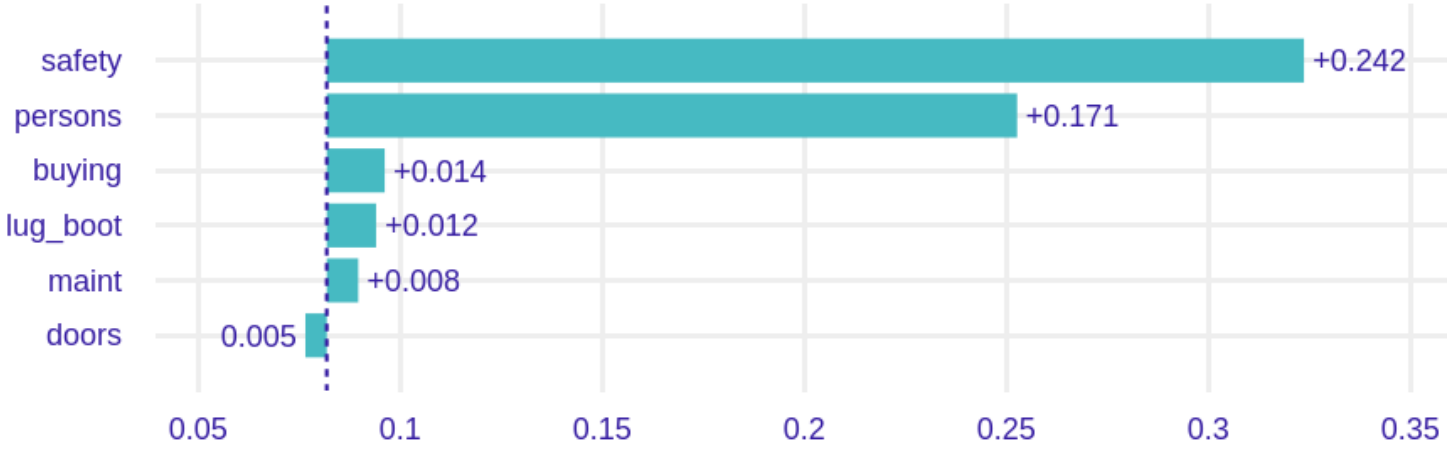


Figure 10: ANN feature importance as computed by DALEX

5.2 Preference analysis

The ANN model is essentially completely opaque. There are no explicit parameters we could use to learn about DM's preferences. However, the feature importance plot (and other visualizations, some of which are presented later on in this report) can give us some insight:

- the order of feature importance is very similar to the one produced by ANN-UTADIS. The only difference is "right-shifting" the 3rd, 4th and 5th most impactful features by 2 (UTA: *maint*>*buying*>*lug_boot*; ANN: *buying*>*lug_boot*>*maint*)
- *safety* and *persons* dominate other features in terms of assigned importance
- very interestingly, *door* is completely irrelevant or even counterproductive. This is a clear example of how a black box model can learn something that makes no sense in the real world.

A fun way to summarise this picture, is to think that the model created an image of someone looking to buy a bus - a safe vehicle, capable of carrying many passengers, and one they are unlikely to have to pay for on their own. Low priority on

lug_boot still makes sense, as well as not really caring about the number of doors - a tourist coach can do fine with one, and it might make counting passengers, or collecting payments from them easier.

5.3 3-alternative analysis

5.3.1 Analytical approach

There is no way to analytically give any estimates on how the output would change, as the created neural network has too many parameters interacting in too complex ways for a human to reasonably handle. The only thing that can be said for certain is that making the same change for *mid* as in ANN-UTADIS (setting *doors*=0.25) would not result in the model deeming it acceptable.

I would wager that similarly as in ANN-UTADIS, changing just a single parameter would not be enough to change the classification of the *best* and *worst* alternatives. However, since *mid* has a *safety* of 0, and this is the most impactful attribute, there probably exists a threshold on it that if *mid* were to exceed, it would make the model change its mind on this alternative's classification. This threshold is probably at least in the neighbourhood of 0.25, but I have no confidence in making more precise guesses.

5.3.2 Space sampling

To concisely show how changing the value of just one attribute would change the result, ceteris paribus analysis was performed. Files with all plots are available in `report/img/extras`. In order not to overcrowd the report, only the plots for *safety* for the 3 alternatives are presented.

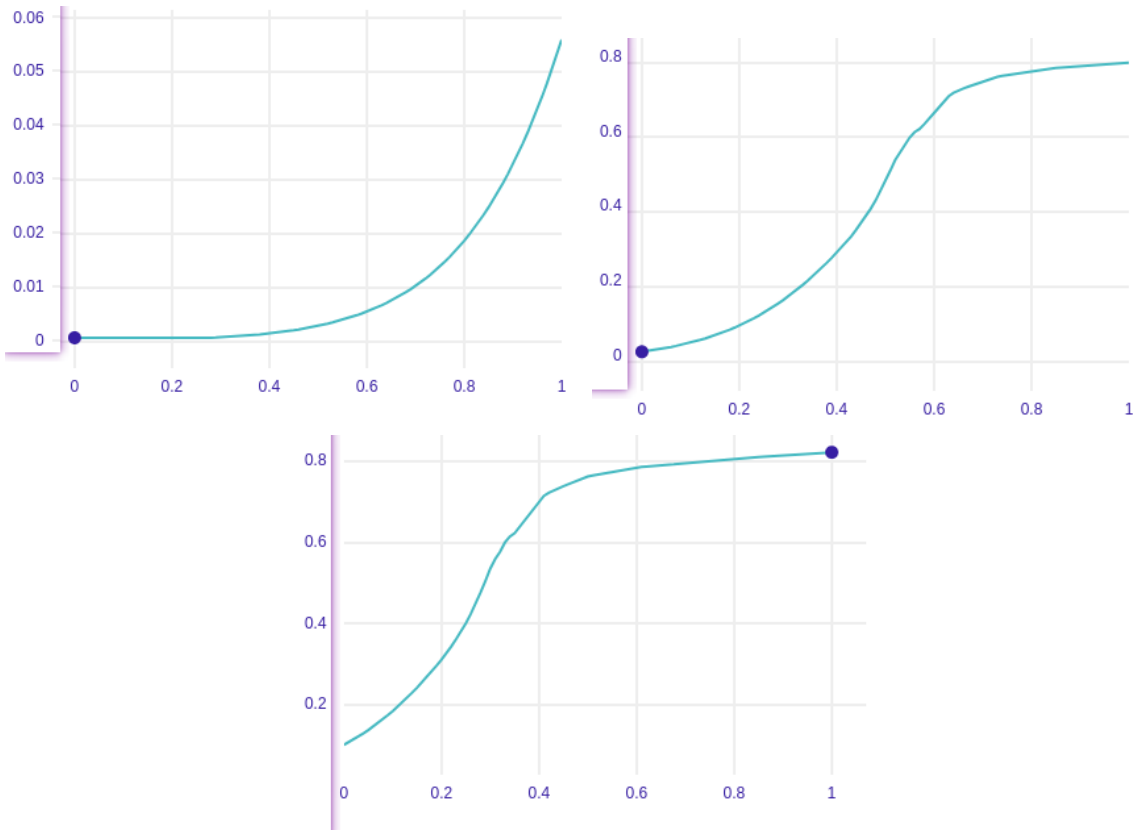
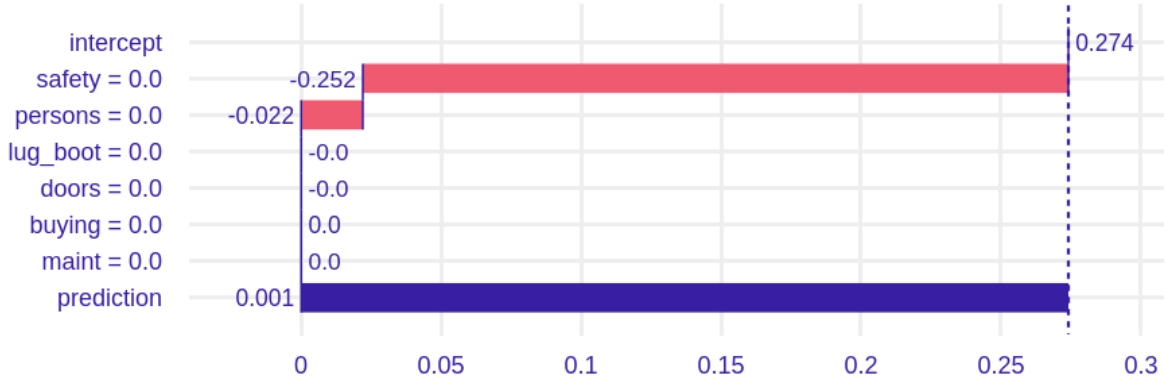


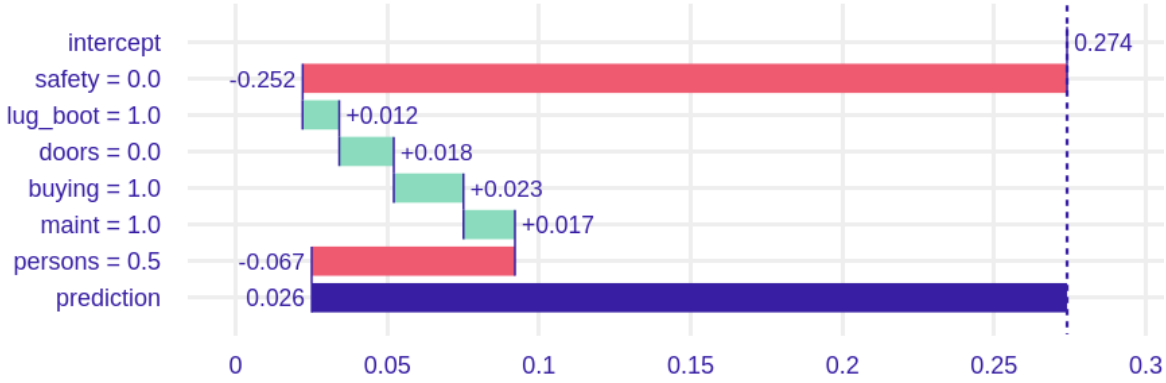
Figure 11: Ceteris paribus analysis for criterion *safety* for (in order, from left to right and top to bottom): *worst*, *mid*, *best*

These plots suggest, that to make *mid* acceptable, setting safety to around 0.5 would be necessary. No amount of change on any other attribute would make *mid* acceptable.

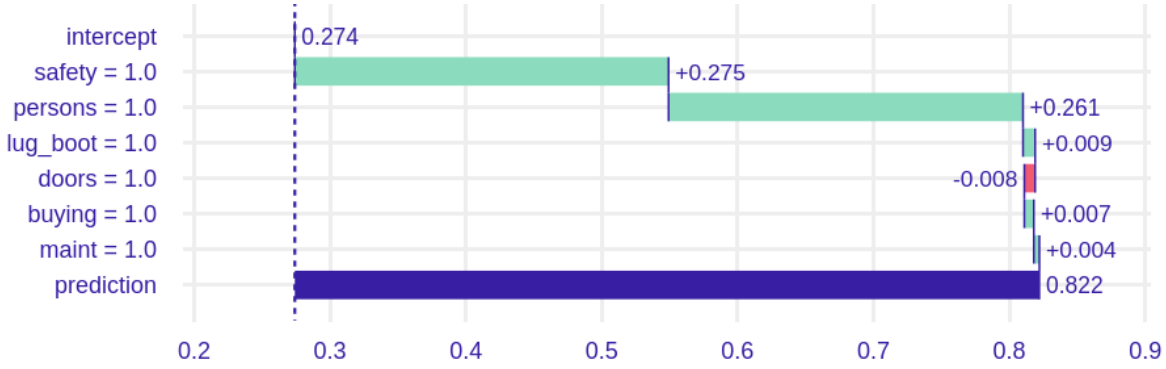
5.3.3 Variable contribution plots



(a) Variable contribution of alternative 1 (worst) in ANN



(b) Variable contribution of alternative 2 (mid) in ANN



(c) Variable contribution of alternative 3 (best) in ANN