Exercise 1: PDP and ICE in case of interactions

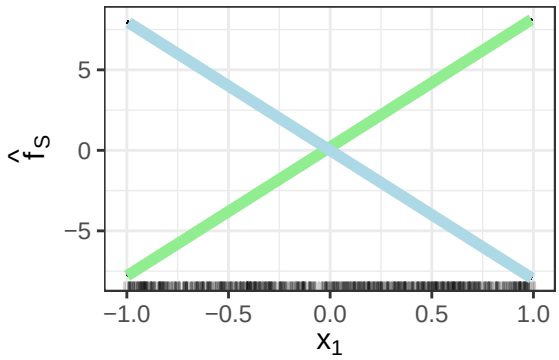
Let us assume that we fitted the following linear regression model with two features:

$$\hat{f}(\mathbf{x}) = \hat{f}(x_1, x_2) = \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_1 x_2 + \hat{\beta}_0. \tag{1}$$

a) Analytically derive the PD function of feature $S = \{1\}$ (with $C = \{2\}$).
Hint: In the lecture slides we derived the PD function for a linear regression model without an interaction term.
Hint: In the end, your solution should include terms like the expected value of X_2.

b) Let us assume that the estimated coefficients in Equation 1 are $\hat{\beta}_0 = 0$, $\hat{\beta}_1 = -8$, $\hat{\beta}_2 = 0.2$ and $\hat{\beta}_3 = 16$. Furthermore, $X_1 \sim Unif(-1, 1)$ is uniformly distributed between -1 and 1, and $X_2 \sim B(1, 0.5)$ origins from a Bernoulli distribution. Compute the exact PD function of feature X_1 using your derived function of a).

c) The following plot displays the ICE curves of X_1. The rugs show the marginal distribution of X_1. Please note that since X_2 is binary, we do not receive n individual ICE curves, but indeed only two unique ones. Derive the conditional expectation functions $\hat{f}_1^{(i)}(x_1)$ of X_1 for group $X_2 = 1$ and for group $X_2 = 0$ using Equation 1 given the estimated coefficients of b). Which color coding (light green or light blue) reflects the group $X_2 = 1$, which one group $X_2 = 0$?



d) Add the PDP you derived in b) to the plot. Use this example to explain briefly why it is advisable to display not only the PDP but also the ICE curves, when visualizing the feature effects of a specific model.

Exercise 2: ALE

Accumulated local effects describe how features influence the prediction of a machine learning model on average. The following tasks guide you to implement ALE from scratch. The associated files are *ale.R* or *ale.py* depending on the programming language you prefer.

a) Since ALE works with bins, we have to select bounds first. Usually the bins are chosen s.t. every bin has roughly the same number of data points. However, to make it more simple, we use the same length for every interval. Complete the function `get_bounds()` to select bounds between the minimum and maximum value of a feature by using `n_intervals`. Those bounds are used for the calculation of ALE in the next tasks.

b) Implement the ALE algorithm from scratch inside the function `calculate_ale()` using the bounds from the previous task and the help of the lecture material. For each interval `k`:

- Select the observations inside the interval (if zero observations are found then return 0 for this interval). In general x_s can be counted as included if it is bigger than the lower bound and smaller equals the upper bound. However, make sure the elements in the first bound are included as well.
- Intervention: Use the relevant observations to create two new datasets:
  - `X_min`: Values at s-th feature position are replaced by the lower interval value z_{k-1}.
  - `X_max`: Values at s-th feature position are replaced by the higher interval value z_k.
- Prediction: Get the predictions for both `X_min` and `X_max`. Compute the differences in predictions of `X_max` to `X_min` and average them.

Use `cumsum` in R or `np.cumsum` in Python to perform the aggregation step and therefore to get accumulated values for each interval value. Finally, if the `center` parameter is set, center each interval by subtracting the average of all interval values
Hint: `mapply` in R or `zip(bounds, bounds[1:])` in Python are a nice trick to iterate over the lower and upper bounds at the same time.

c) The function `calculate_ale()` returns both the used bounds and the ALE data. To use those data with `ggplot2` (R) or `matplotlib` (Python), you have to get the corresponding interval centers. Implement the function `prepare_ale()`, which should call `calculate_ale()` first and then compute the averages/centers per interval. In the end `prepare_ale()` should return the centers plus the corresponding ALE values - either as a `data.frame` (R) or two vectors (Python).

Exercise 3: Categorical ALE

In this exercise, we use the German Credit dataset (*datasets/credit.csv*) and a random forest to predict whether a customer is a high or low risk for a bank. The dataset could be found in *credit.csv*. The following table gives an overview:

	name	type	mean	nlevs
1	credit_risk	factor		2
2	age	integer	35.55	0
3	amount	integer	3271.26	0
4	credit_history	factor		5
5	duration	integer	20.90	0
6	employment_duration	factor		5
7	personal_status_sex	factor		4
8	purpose	factor		10

Since the decision of whether a person gets a loan can have serious implications on the person's life, banks are subordinate to regulations and must disclose the underlying mechanism of their used model to authorities. Since looking on the single trees is not feasible to uncover the internals of a random forest, (model-agnostic) interpretation methods should help to unfold the underlying mechanisms and to explain specific decisions.

a) Since the regulations require that the model does not discriminate against certain demographic groups, you want to evaluate the feature effect of `personal_status_sex` of the fitted forest model. Your colleague provides Figure 1. Interpret the PDP.

b) You also request an ALE plot from your colleague. Give reasons why and in which situations inspecting ALE plots besides PDPs is advisable?

c) Because the ALE models accumulates effects in a specific direction, the feature values must have an order by definition. However, there is no natural order to categorical/nominal features like `personal_status_sex`. In order to derive an artificial order, Molnar 2022 (Chapter 8.2)[1] proposes to order the categories of the features x_s according to their similarity based on other/remaining features $x_c \in X_C$. Read the corresponding subsection in Molnar 2022.

In detail, the approach has the following steps:

[1] https://christophm.github.io/interpretable-ml-book/ale.html

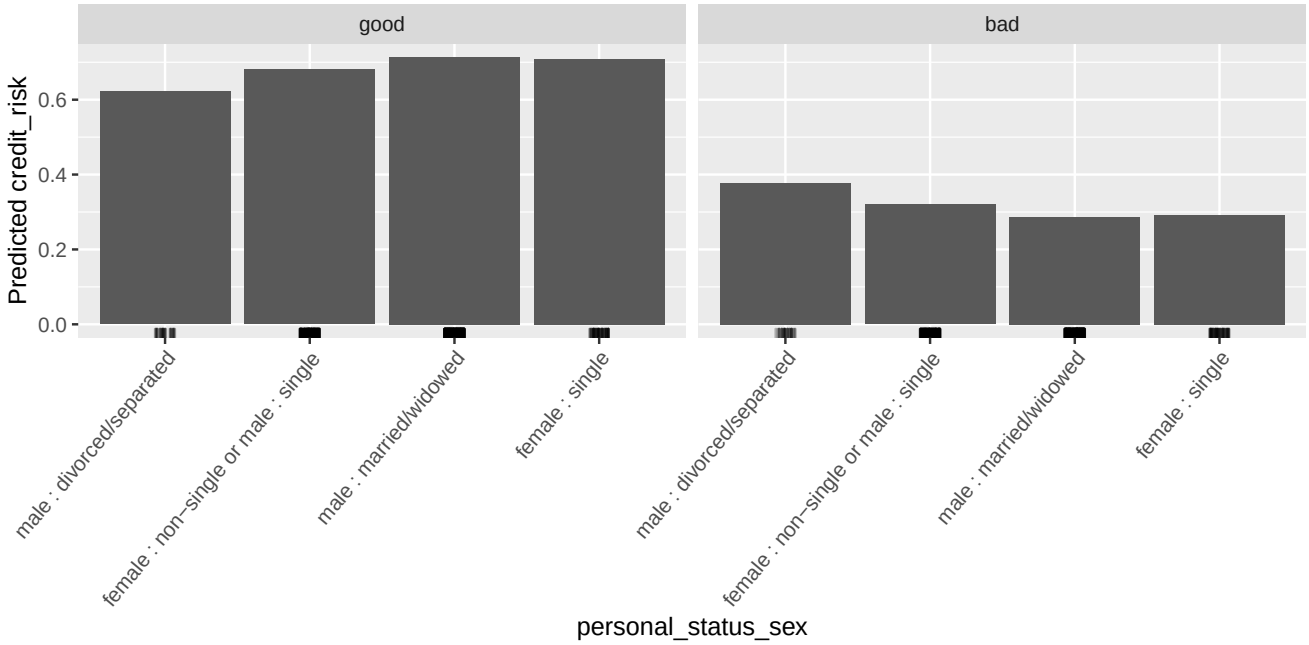


Figure 1: PDP of personal_Status_sex

1) For each feature $x_c \in X_C$ do the following:
   i. Get the conditional empirical distributions of x_c given the categories of x_s:
   For numeric feature x_c, estimate the cumulative distribution function for each category of x_s separately and derive its values at predefined points (e.g., at deciles of x_c). For categorical feature x_c derive the relative frequency tables for each category of x_s separately.
   ii. Compute the pairwise distances of distributions for each pair of categories of x_s:
   For numeric feature x_c, the distance is equal to the absolute maximum point-wise distance of the two empirical distribution functions. For categorical feature x_c, the distance is equal to the sum of the absolute difference of both relative frequency tables.
2) end for
3) Sum up the pairwise distances of the distributions over features x_c in X_C for each class pair of x_s.
4) Reduce the resulting distance matrix to a single dimension using multi-dimensional scaling.
5) Order the categories of x_s according to the obtained similarity values.

Your colleague already implemented some parts of the algorithm. Depending on your preference, you can access either the R code in *catale.R* or the Python code in *catale.py*. Your colleague needs your help with implementing the computation of the distance of distribution for a **categorical** feature x_c in `get_diff_cat()`.

- Have a look on the function `order_levels()` and `get_diff_numeric()`. Locate the steps of above's algorithmic description in the underlying code. Examples of code execution are given at the script's end.
- Implement the missing rows in `get_diff_cat()`.
  Hint: The command `expand.grid(levels(feature.j), levels(feature.j))` creates the first two columns in your resulting dataset - the pairwise class combinations for x_s.
- Test your function with the provided example code (end of script). Your returned `data.frame` for `personal_status_sex` as x_s and `employment_duration` as x_c, should look similar to this:

class1	class2	dist
male : married/widowed	male : married/widowed	0.00
female : non-single or male : single	male : married/widowed	0.22
male : divorced/separated	male : married/widowed	0.19
...	...	...

- How did your function order the levels of `personal_status_sex`? Does this order make sense to you?
- **Bonus:** Critically assess whether PDP/ALE alone replace a complete fairness audit.