

**Solution 1:**

- a) Permutation Feature Importance compares the performance of the model on the original data with the performance on perturbed data, where the dependence of the variable of interest (let's call it  $x_j$ ) with the target  $Y$  variable is broken.

In order to break the dependence of  $x_j$  with  $y$ ,  $x_j$  is replaced with a permuted version  $\tilde{x}_j$  which is independent of the target.

However, by permuting the variable we do not only break the dependence with  $y$ , but also with all other covariates. As such, we may create unrealistic observations.

For example, time of the year may be dependent with the highest temperature on a day. If we resample the variable time of the year independently of the temperature high, we may create observations where time of the year is winter and temperature high is 40 degrees celsius.

- b) If a feature anyway independent of its covariates ( $x_j \perp x_{-j}$ , then PFI does not extrapolate to unseen regions.

Intuitively, the reason is that no dependence with the covariates is broken, since there were not dependencies between  $x_j$  and the remaining variables  $x_{-j}$  to begin with.

- c) Pseudocode reading in data, estimating a regression model and calculating the MSE

---

**Algorithm 1** getting started

---

```
1: df ← read in dataset extrapolation.csv
2: X, y ← features, target
3: X_train, X_test ← training data, test data
4: model ← linear regression model on X_train
5: return MSE of model predictions
```

---

- d) First, we implement PFI. Here we implement the methods generically, such that the perturbation mechanism can easily be replaced. Therefore, we make use of `f(*args,**kwargs)` and `f(...)` in Python and R, respectively.

Pseudocode of `pfi_fname()`

---

**Algorithm 2** `pfi_fname()`

---

**Require:** `fname`: feature of interest name

**Require:** `predict`: prediction function

**Require:** `score`: performance metric

**Require:** `X_test`: data for the evaluation

**Require:** `y_test`: respective labels

**Require:** `...`: further arguments (which are ignored)

```
1: X_test_perturbed ← copy X_test and randomly permute column containing feature fname
2: performance ← score(y_test, predict(X_test_perturbed)) - score(y_test, predict(X_test))
3: return performance
```

---

Pseudocode of `fi_naive()`

---

**Algorithm 3** `fi_naive()`

---

**Require:** `perf_pert`: function that returns performance for some perturbation.

**Require:** `predict`: prediction function

**Require:** `score`: performance metric

**Require:** `X_test`: data for the evaluation

**Require:** `y_test`: respective labels

**Require:** `...`: further arguments (which are ignored)

```
1: for fname in columns of X_test do
2:   imp ← fi_fname(fname, predict, score, X_test, y_test, ...)
3:   results ← extend by imp
4: end for
5: return results
```

---

Pseudocode of `n_times()`

---

**Algorithm 4** `n_times()`

---

**Require:** `n`: number of repetitions

**Require:** `method`: feature importance method.

**Require:** `args`: all further arguments that are required for the method

**Require:** `return_raw`: Whether only the aggregation (mean, std) or also the raw results are returned

```
1: for i in n do
2:   tmp ← method(args)
3:   results ← extend by tmp
4: end for
5: mean_fi ← mean of results
6: std_fi ← std of results
7: if return_raw equals True then return mean_fi, std_fi, raw_results
8: else if return_raw equals False then return mean_fi, std_fi
9: end if
```

---

Now we apply the method to our model and dataset.

---

**Algorithm 5** application

---

```
1: nr_runs ← 10
2: fis_mean, fis_std ← n_times(nr_runs, fi_naive, pfi_fname, model_prediction_function,
    mean_squared_error, X_test, y_test, return_raw=False)
3: plot bar chart
```

---

e)  $X_3$  is the most important feature, with  $X_1$  and  $X_2$  sharing the second place. PFI considers  $X_4$  to be irrelevant. According to the PFI interpretation rules, without further assumptions about the data, we know that

- $X_1, X_2, X_3$  are used by the model for its prediction
- $X_1, X_2, X_3$  are dependent with  $Y$  and/or dependent with the covariates
- $X_4$  may be independent of  $Y$  and covariates and/or not used by the model. We only know that it is not both dependent and used by the model.

Bonus: If we would additionally analyze the data we find out that  $X_1, X_2$  are dependent but  $X_3$  is independent of all covariates.

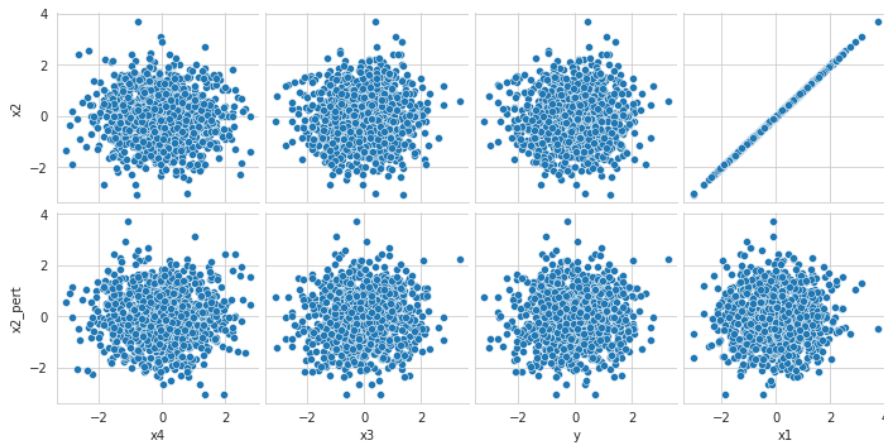
As such, we could further conclude that  $X_3$  is dependent with  $Y$  (but do not know for  $X_1, X_2$  without looking into the data).

f) Correlation matrix:

	$x_1$	$x_2$	$x_3$	$x_4$	$y$
$x_1$	1.0***	1.0***	0.03	-0.05	0.02
$x_2$	1.0***	1.0***	0.03	-0.05	0.02
$x_3$	0.03	0.03	1.0***	-0.0	1.0***
$x_4$	-0.05	-0.05	-0.0	1.0***	0.0
$y$	0.02	0.02	1.0***	0.0	1.0***

Our interpretation was correct.

- g) If we know the dependence structure of the covariates we can infer whether or not the PFI value is nonzero due to a dependence with the covariates or not. In our example we now know that  $x_3$  is independent of its covariates, so we hypothesize that  $x_3$  is actually dependent with  $y$  (and would perform importance tests as a consequence). Since  $x_1, x_2$  are dependent, for those variables we cannot infer anything about the dependence with  $y$  with the covariates dependence structure and PFI alone.
- h) We use the `extrapolation.csv` dataset. We create a pairplot showing the pairwise scatterplot of the original feature as well as the corresponding perturbed variable with all remaining feature variables (and potentially  $y$ ).



We can see that the strong correlation of  $x_2$  with  $x_1$  (top row) is broken after permutation (bottom row). All other pairwise (in)dependencies are unchanged. Note: We only assess pairwise, unconditional dependencies. Without assumptions about the data, we cannot know whether further conditional dependencies with the remaining covariates were broken.