Creator **Olusanmi Hundogan** Created **Jan 09, 2024, 14:03** Last updated **Jan 12, 2024, 01:32**

Files

- notebooks/analyze_missing_value_imputers_equilibrium.py
- experiments/experiment_missing_value_imputers_equilibrium.py
- imputer design inspiration:

Motivation

This experiment acts as a preliminary study for the experiment which compares different imputation methods. For this experiment we implemented an imputation method called the EquilibriumImputer (formerly called: Sorin's Crazy Monte-Carlo method). The method imputes values by iteratively filling the features over multiple runs across all features leading to an eventual convergence. We implemented two variations of this method:

- EquilibriumImputer: Imputes every feature using the other features from the previous full feature iteration
- FastEquilibriumImputer: Imputes every feature and updates them directly without waiting to finish the full feature iteration.

The convergence criterion which captures the distributional change of the features was reached if all features fell below the threshold, as described by the following formula from the paper linked above ("MCMC_imputation_paper_good.pdf").

The convergence of the imputed data across iterations of the imputation process is analysed by studying how both the median, M_y, and the interquartile range, IQR_y, of each imputed variable, y, change across two consecutive iterations, t and $t-1$, as follows:

$$\sqrt{\left(\begin{bmatrix} M_y^{(t)} \\ IQR_y^{(t)} \end{bmatrix} - \begin{bmatrix} M_y^{(t-1)} \\ IQR_y^{(t-1)} \end{bmatrix}\right)' \left(\begin{bmatrix} M_y^{(t)} \\ IQR_y^{(t)} \end{bmatrix} - \begin{bmatrix} M_y^{(t-1)} \\ IQR_y^{(t-1)} \end{bmatrix}\right)} \quad (15)$$

There are additional stopping criterions like whether the differences between former iteration and current converge, the majority of differences increase, or a maximum number of iterations was reached.

This method puts more emphasis on the inference/transform step, rather the training/fit step.

Design

Within the experiment we explored different thresholds for the convergence criterion. The values where [0.01, 0.001, 0.00001]. We failed to test 0.0001 as well. However, as seen below, the convergence threshold does not appear to influence performance.

We also implemented a mechanism that skips columns in subsequent iteration, should they have converged prior. We kept the diff_threshold at 0 (effectively turning it off) as it appears to correlate with the convergence threshold. We stopped early if 100 full iterations were reached or 60% of feature differences increased.

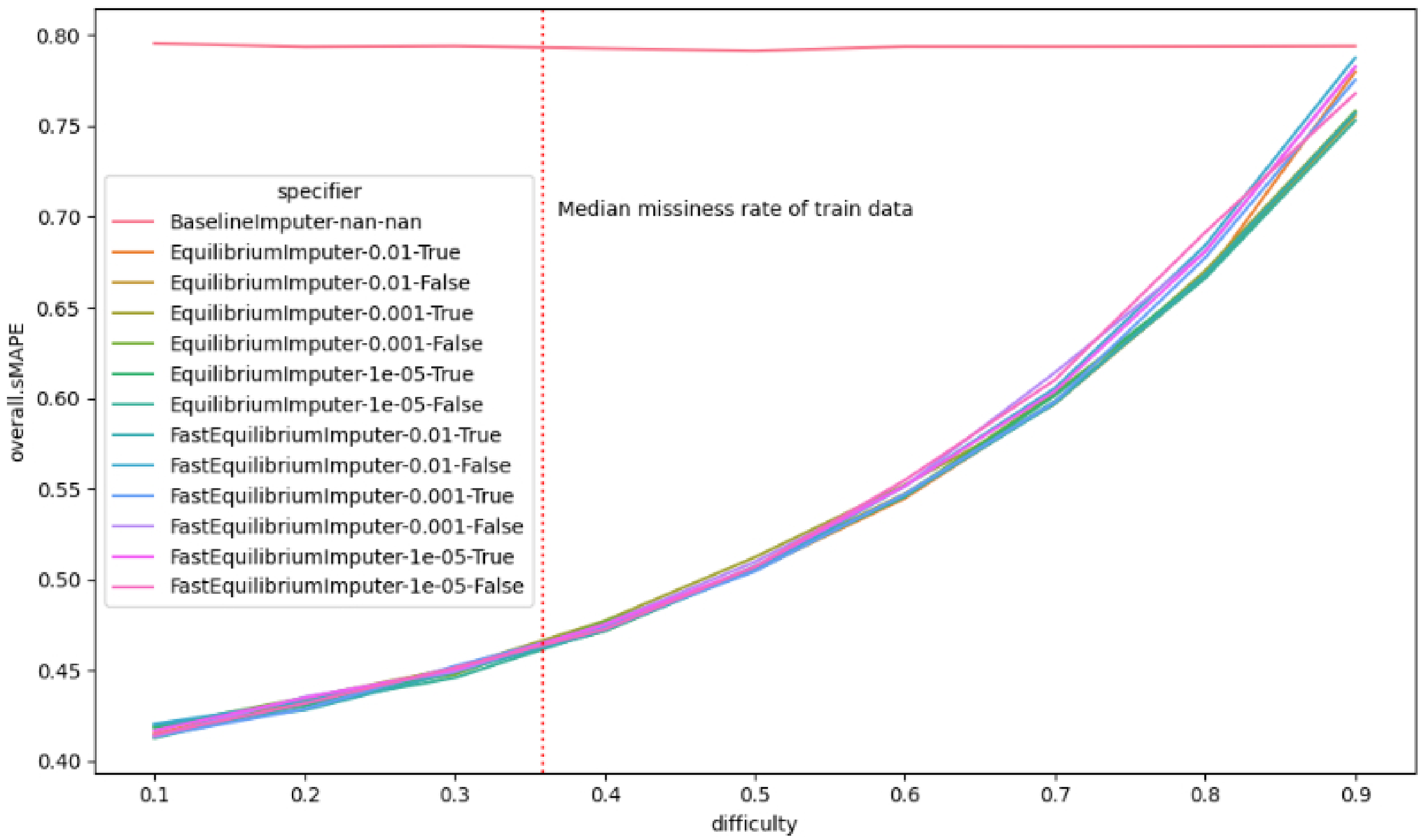
We evaluate the imputer by removing values from a dataframe gradually and measuring the sMAPEof reconstructing the values. We test difficulties from 0.1 to 0.9 in even steps. A difficulty of 0.1 corresponds to the artificial removal of 10% of known values. (The number is not completely accurate because the 10% applies to the dataframe, but the dataframe already contains missing values.)

We also only impute on features that have a rate of missingness below 30%. After applying this feature selection 50% of the data rows have a missingness of below 10%.

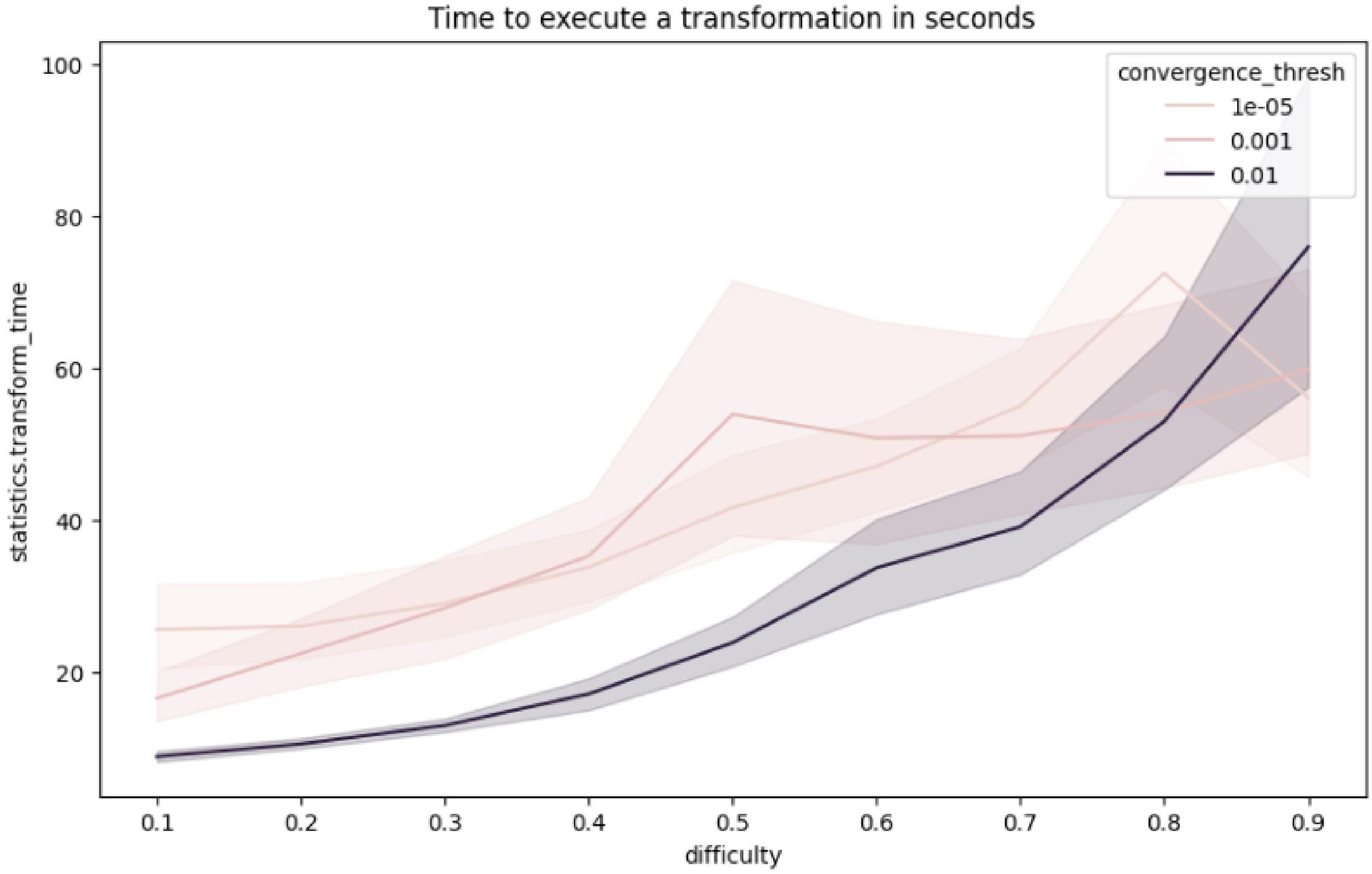
We also measure the time of imputation in seconds and the amounts of completed full iterations.

Results and Insight

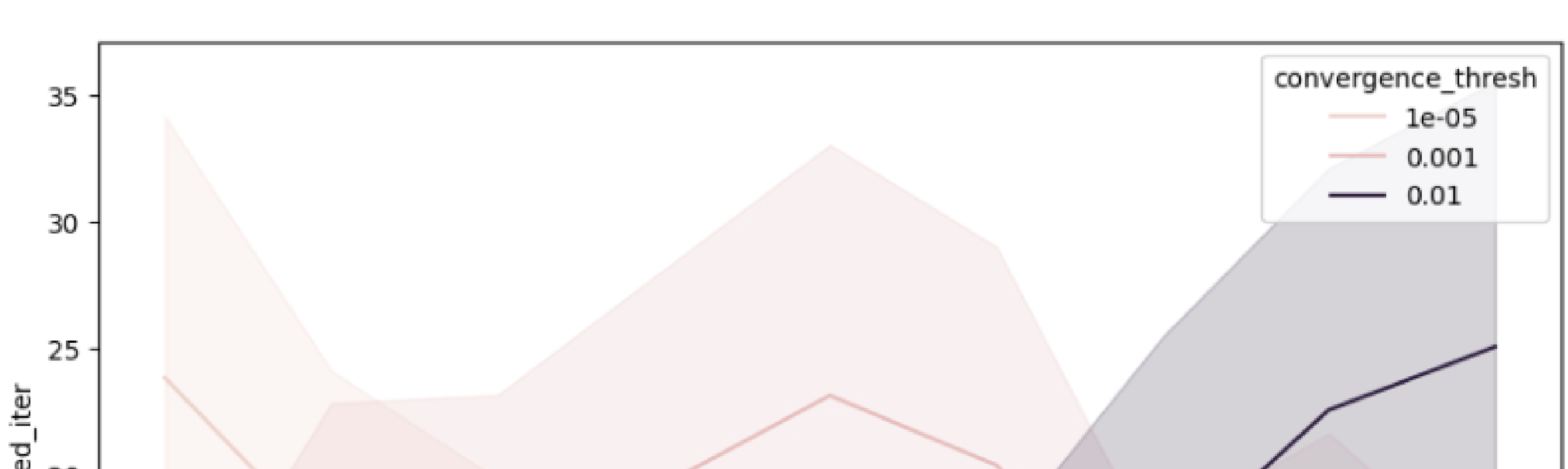
The plot shows all the different configurations tested across all difficulty levels. All configurations perform roughly the same across all difficulties. This result indicated, that the configuration itself has no effect on the performance. However, there are effects with respect to the time as seen in the figures below.



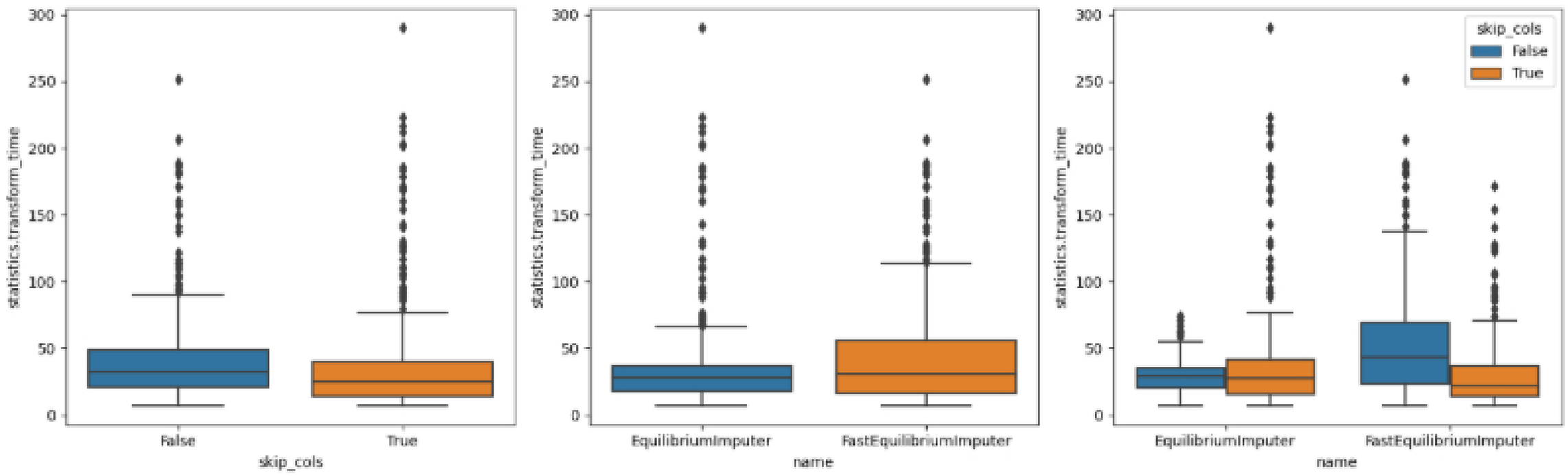
The transformation time increaseswith difficulty, which is also in line with the difference in convergence thresholds: the highest threshold (0.01) has the most pronounced behaviour.



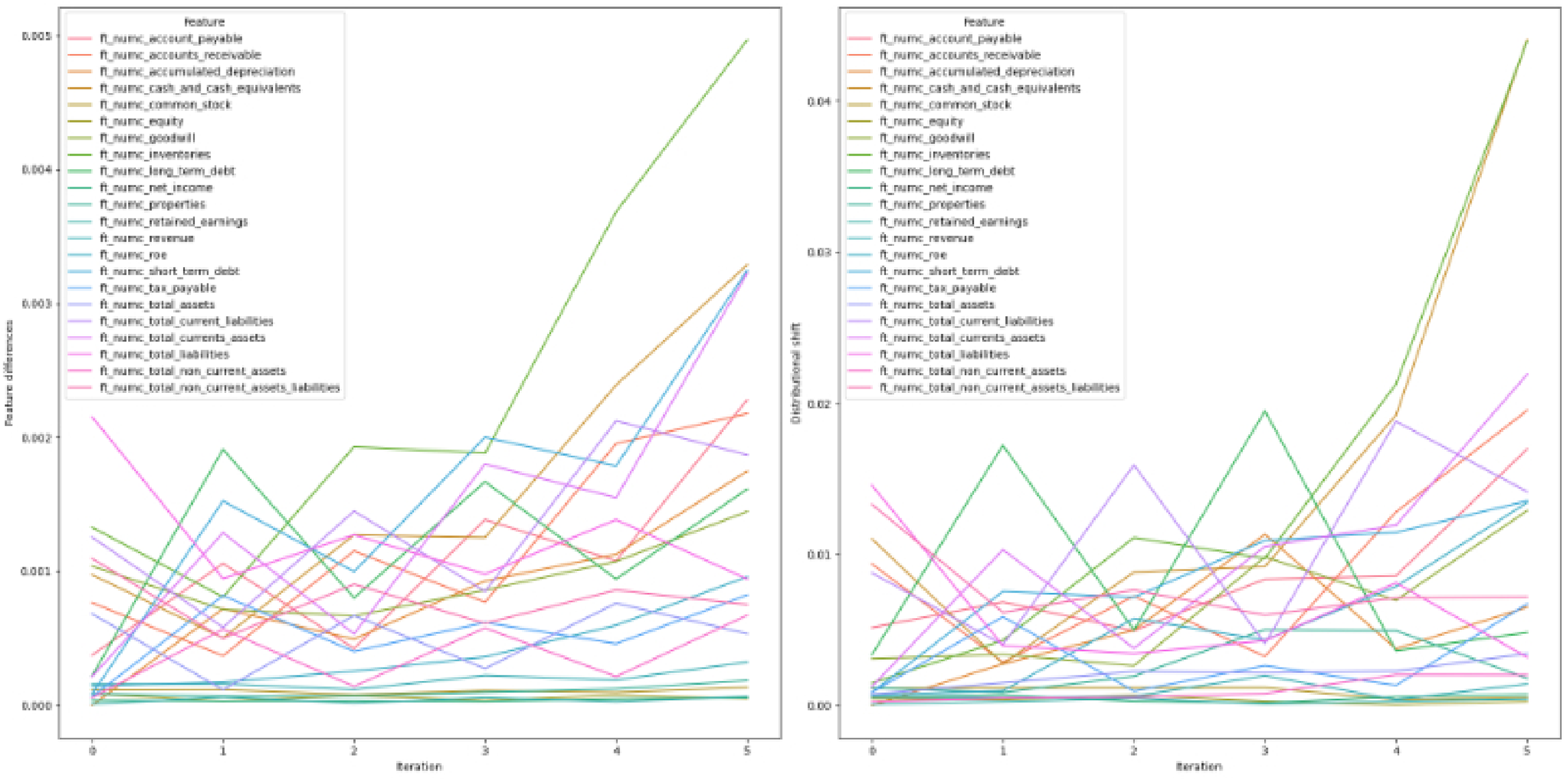
In terms of iteration completion we again see a very clear behaviour for 0.01 but lower convergence thresholds show a more erratic behaviour. This is possibly due to the interference of other stopping criteria.



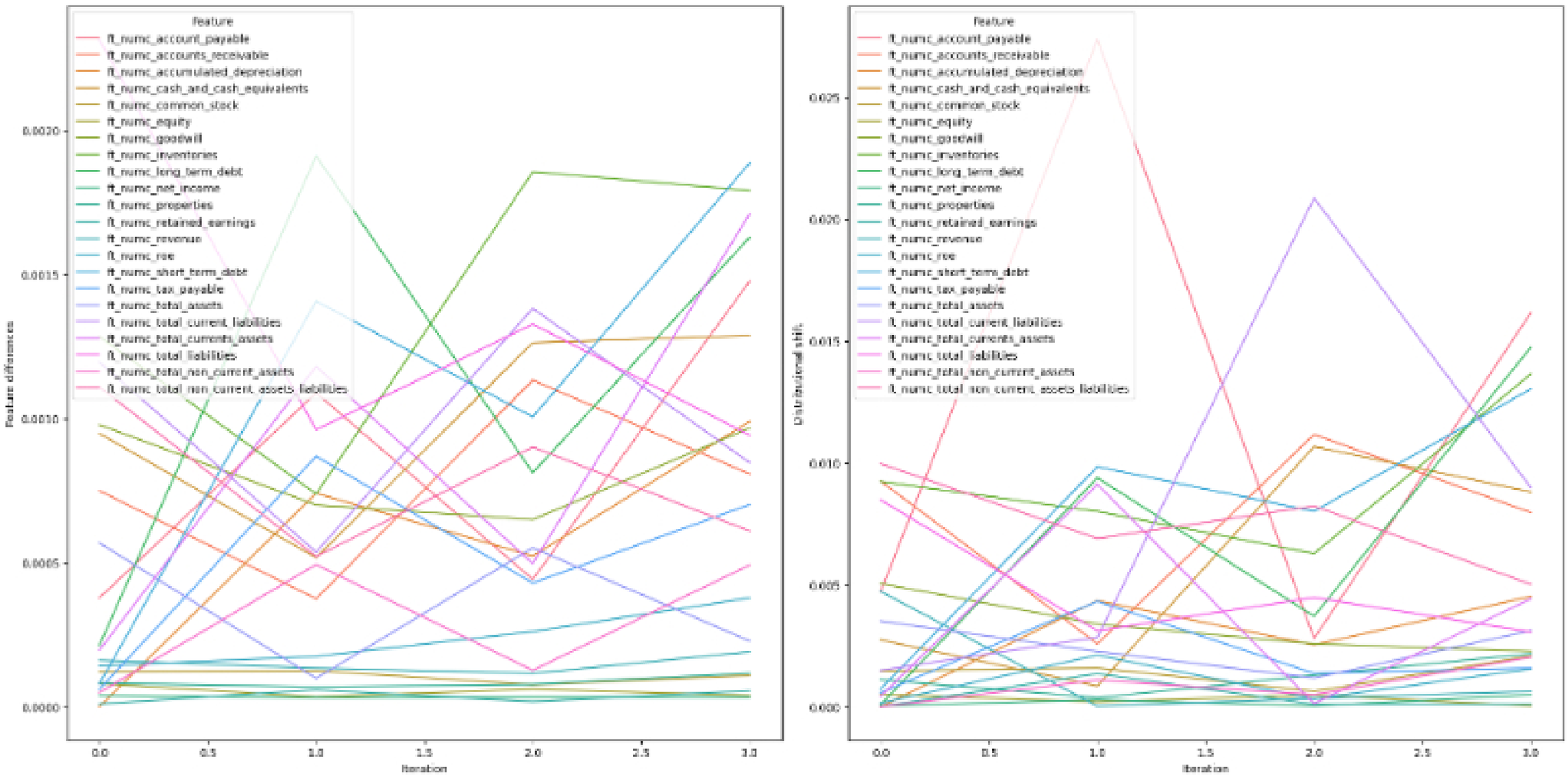
The last figure shows the factors of the columns skipping mechanism and the algorithm variant and their interaction. The fastest model appears to be the EquilibriumImputer without skipping mechanism. However, this is because this configuration often stops early due to the feature differences increasing.



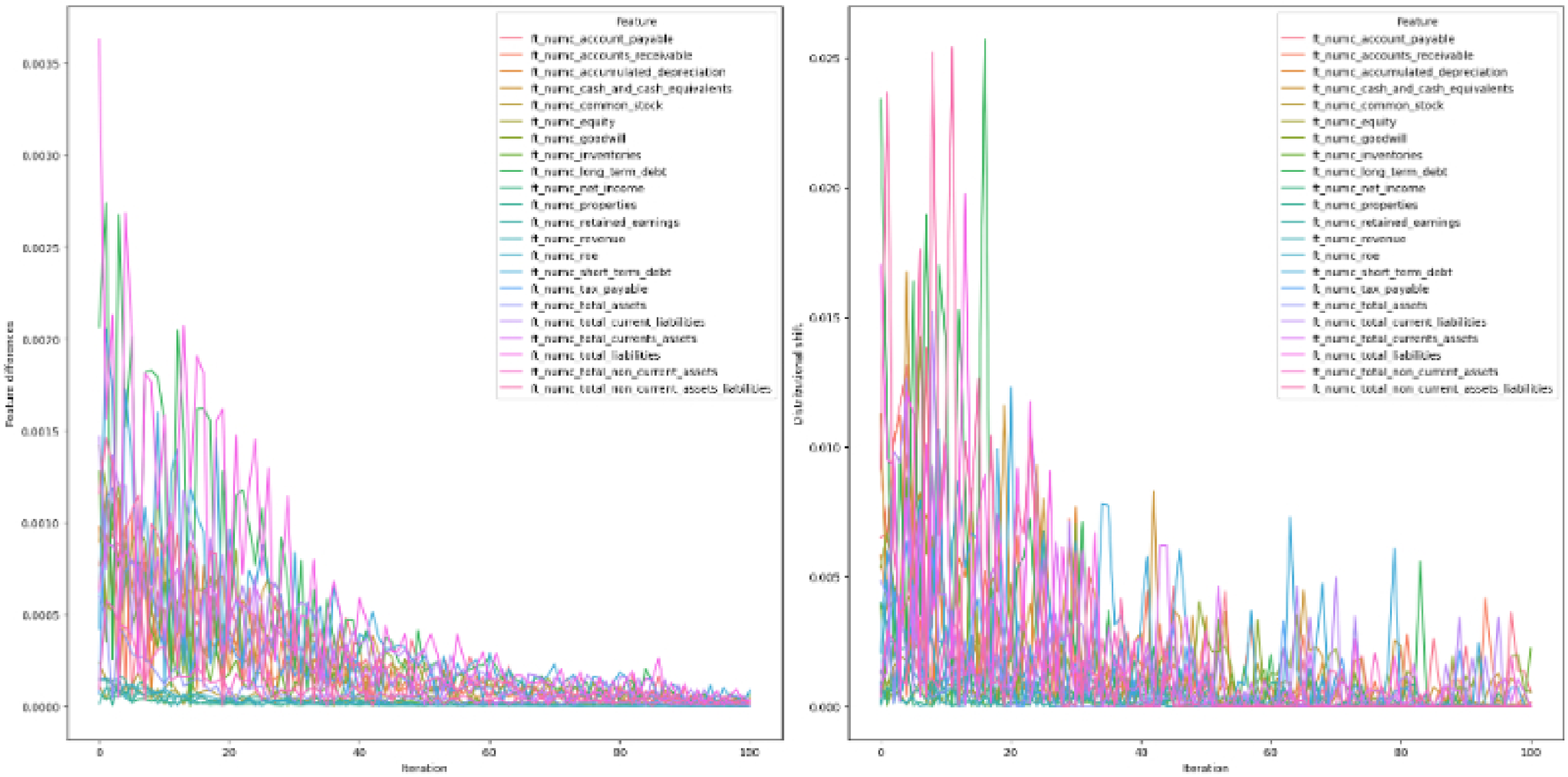
We can see the diverging characteristic of the differences in this plot which shows the EquilibrumImputer without skipping mechanism. (Left shows differences of the features compared to the last iteration and right shows the convergence criterion values)



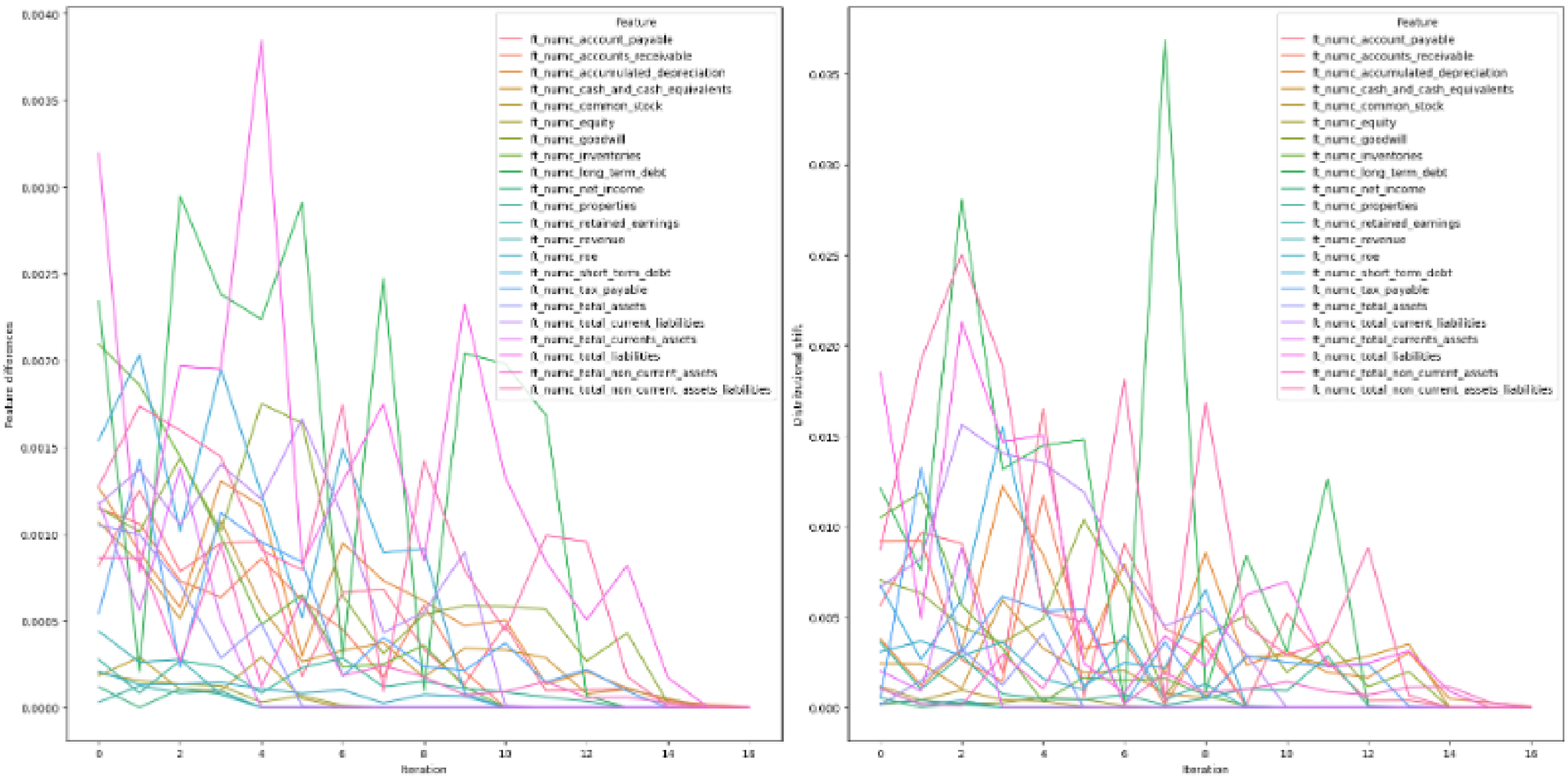
Same holds for the case in which the skipping mechanism is activated on EquilibriumImputer.



If we do not skip in the FastEquilibriumImputer method, we see a very clear convergence pattern.



This pattern remains but the transformation stops significantly earlier if columns are skipped for the FastEquilibriumImputer.



Decision

Update : 09.01.2024

We emphasize, that all the configurations perform very similarly. Therefore, the decision is governed by temporal and heuristic considerations. We chose the FastEquilibriumImputer with skipping because it shows a converging pattern and remains reasonably fast.

The next step is comparing one of the EquilibriumImputer variants with the previously competing imputers.

Update : 12.01.2024

The method will not be included as running the experiment takes a significant amount of time and that makes this approach infeasible for production .