LIME vs SHAP
Comparing ML explanation techniques

Ethics in Artificial Intelligence 2023-2024

Andrea Terenziani
andrea.terenziani@studio.unibo.it

Abstract

LIME[1] and SHAP[2] are two approaches to extracting feature importances for any Machine Learning model, the former using a proxy explainable model (like a decision tree) and the latter using a game-theory driven method based on Shapley values.

The idea of this project is to solve a classification task with different models, computing a LIME and SHAP explanation for each, then compare those with a "baseline explanation" obtained directly from a decision tree or random forest model. This comparison aims at finding possible links between a model's performance and its explanations' similarity to the baseline, and at determining how "easily", if ever, LIME and SHAP independently compute similar importances for a given model.

1 Base task

1.1 Dataset

As stated previously, the models were trained to solve a binary classification problem, where the data consisted of attributes of 136429 industrial devices, each characterized by six features (see table 1).

The class was encoded as `0` (no failure) or `1` (failure). The main issue with the data is its extreme *imbalance* between the two classes, with a ratio of around 62 to 1 "in favor of" the 0 class. This was addressed (mostly) by:

1. resampling the minority class
2. choosing an appropriate metric for tuning

The resampling specifically consisted in *oversampling* the minority class to reach a minority-majority ratio of 1 to 10; while this clearly didn't fully balance the data, duplicating the relatively few minority datapoints even was avoided for fear of overfitting. Undersampling the majority was also discarded because, in this case, it would mean removing a large number of rows to achieve the target ratio.

1.2 Models

All estimators were first run using default parameters, to achieve a baseline performance, then tuned with cross validation. All this was done using the `scikit-learn` python package [3], with the cross validation being performed using the `GridSearchCV` and `StratifiedKFold` classes.

Attribute	Datatype	Description
Type	`String`	Type of product/device (possible values: "L","M","H")
Air Temperature	`Float`	Air temperature (Kelvin)
Process Temperature	`Float`	Production process temperature (Kelvin)
Rotational Speed	`Int`	Speed in RPM
Torque	`Float`	Torque in Nm (Newton Meter)
Tool Wear	`Int`	Time unit needed to wear down the product/tool
Machine Failure	`Int`	Failure binary feature (*class attribute*)

Table 1: Features

The following models were tested:

- Decision Tree
- Random Forest
- Linear Perceptron
- AdaBoost [4]
- K-Nearest Neighbor
- Gaussian Naive Bayes
- Linear SVM (referred to here as *SGD*)

1.2.1 Training results

The finetuning process revealed that the metric to be maximized, when searching for optimal hyperparameters, was the macro average of the recall for each class (identified as `"recall_macro"` by sklearn) - this is because it "requires" a model that correctly classifies as many classes as possible, giving more importance to the minority. It was therefore used as a metric to gauge the quality of each classifier. The results can be seen in figure 1, showing this metric for each class before and after tuning, and in table 2, ranking each model according to the macro recall achieved for the minority class. Using such metric, we can determine the Random Forest classifier as the best performing.

2 Explanation methods

2.1 LIME

LIME[1] (*Local Interpretable Model-agnostic Explanations*) is an XAI algorithm meant to compute the weight a machine learning model has given to each feature of a datapoint when computing its output. This is done by training a "natively" explainable model (such as a decision tree) on inputs generated by slightly perturbating a given observation, with the idea being that the complex class boundary found by the initial model would become linear in a small enough neighborhood of each input (see figure 2).

	Class 1 recall	Recall macro average
Random Forest	84%	89%
SGD	80%	80%
Perceptron	72%	79%
Decision Tree	71%	83%
AdaBoost	55%	77%
KNN	45%	72%
Gaussian NB	34%	66%

Table 2: Final performances

2.2 SHAP

SHAP (*SHapley Additive exPlanations*) [2] is an alternative to LIME based on *Shapley values*, a concept borrowed from cooperative game theory. The idea is to estimate how much each feature contributed to the model's output by estimating how much it would have changed if it hadn't been present in the data; this is also (theoretically) done over all subsets of features for a given datapoint in order to account for all possible interactions between the attributes. In practice, since the shape of a model's input can't change, the features of an observation are "removed" by replacing them with random values, based on their respective distributions observed in the training dataset.

3 Experiment

As said before, the goal of this project was to compare the results of applying LIME and SHAP to various models trained for the same task. While it is obvious that these algorithms produced different weights for each feature, it was still interesting to ask:

1. with a given explainer algorithm,
   - how similar are the average results for each model?
   - is there a relation between a model's final performance (e.g., its average recall for the minority class) and the estimated importance it gave to each feature?
2. two of the models used, the decision tree and random forest, are "natively" explainable using the Gini impurity metric - for each, is the distribution of these weights similar to that computed via LIME or SHAP?

To answer these, the two algorithms were tested using a subset of 200 observations from the testing dataset. Considering the weights computed for the minority class 1, these were then averaged to obtain a kind of "description" of a model. These results are summarized in figures 3 and 4, each showing, for each model and feature, the weight computed for a certain sample.

3.1 Results

3.1.1 Question 1

Figure 5 shows the average feature weights computed by the two explainers for class 1 - what is noticed immediately is that:

- LIME averages are a lot more similar between different models than their SHAP equivalents;

- SHAP averages tend to be almost exclusively positive.

For each algorithm, the seven histograms are compared by treating them as six-valued vectors and computing their respective cosine distances, which are then normalized between 0 and 1. Figure 6 shows these differences: for both LIME and SHAP, the SGD model is remarkably distant from the rest, while the decision tree, random forest, AdaBoost, and KNN models seem to produce very similar explanations. The perceptron and Gaussian naive Bayes models also show some distance from this "main group", though it is more evident using SHAP values. Since the SGD model was the second best performing, this could mean that it learned and applied a more original interpretation of the data.

3.1.2 Question 2

While a set of seven models is probably a rather small sample to test this, this second part of the 1st question is still worth investigating, since it could also show which, if any, of the two algorithms is better at "detecting" a model's performance.

The result of this comparison is shown in figure 7; on the vertical axis are the average class 1 recall values, as shown in the first column of table 2, while the feature importances are on the horizontal axis. While the SHAP plots (figure 7b) don't seem to show simple nor consistent correlations, from those for LIME (figure 7a) two trends seem to appear:

- for `Type`, `Air temperature` and `Process temperature`, performance *drops* sharply after a certain *negative* threshold
- for `Torque` and `Tool wear`, performance *increases* sharply after a certain *positive* threshold

In the first case, it could mean that a model should *disregard* those features by some degree, while it should "pay" some minimum *additional* attention for the two of the latter case.

3.1.3 Question 3

Tree-based models from the `scikit` library, in this case `DecisionTreeClassifier` and `RandomForestClassifier`, can easily estimate the importance given to each attribute using the `feature_importances_` attribute. From the documentation:

> *The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature*

The criterion is the metric to be minimized by the feature used to split each node of the tree; the one selected during fine tuning turned out to be the *Gini impurity* metric: it measures how often a randomly chosen element of a set (in this context, the set of values of a certain feature) would be misclassified if it were labeled randomly and independently according to the distribution of classes in the set.

Figure 8 shows the result of this comparison, with the Gini and SHAP importances being fairly similar, especially in showing a stronger impact for the first three features, while the LIME values are completely different.

4 Conclusion

This experiment revealed notable differences between LIME and SHAP in interpreting feature importance across various models. LIME produced more uniform feature weights between models, suggesting that it might be less sensitive to specific model architectures, while SHAP showed greater variability and mostly positive weights, indicating potentially higher sensitivity to nuanced patterns captured by the models. The distinct behavior of the SGD model, which was notably different in its feature importance distributions yet performed well, could imply that it learned unique data relationships that LIME and SHAP captured differently. The consistent trends in LIME's outputs, particularly the observed performance changes linked to specific feature thresholds, suggest that LIME might better capture certain performance-related feature behaviors. This could be due to LIME's localized, linear approximations, which simplify complex models but may miss deeper feature interactions that SHAP, with its game-theoretic approach, considers. The closer alignment of SHAP with Gini-based native feature importances in tree-based models supports SHAP's capacity to reflect deeper, non-linear relationships.

References

[1] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[2] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitányi, editor, *Computational Learning Theory*, pages 23–37, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

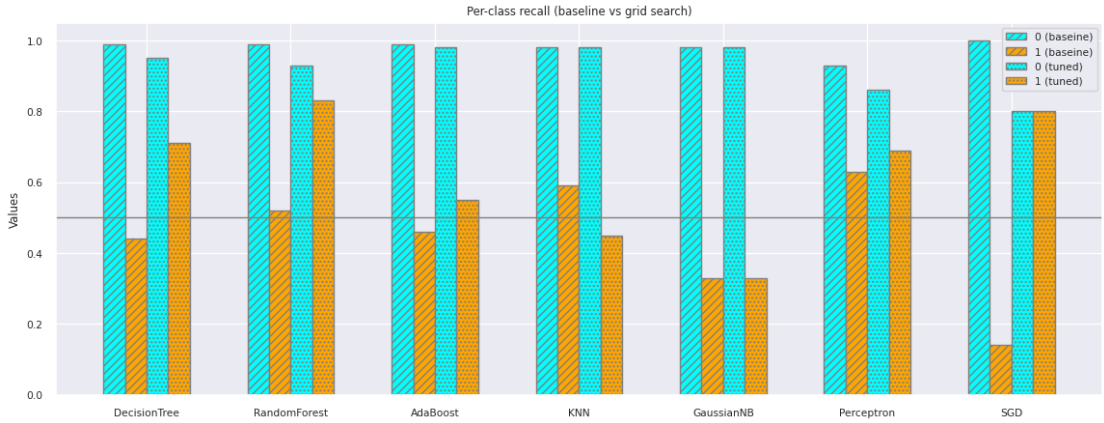


Figure 1: Recall macro average before and after optimization

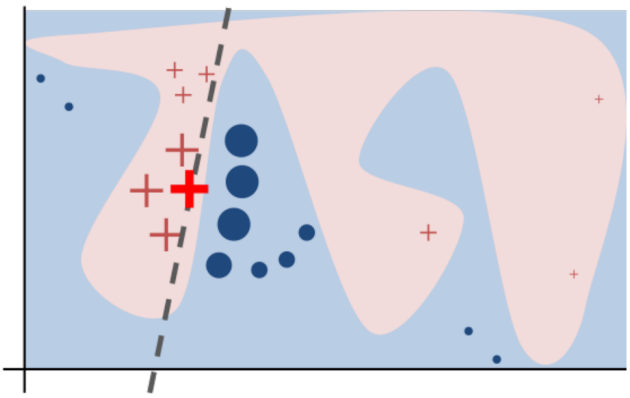


Figure 2: The black-box model's complex decision function f (unknown to LIME) is represented by the blue/pink background, which cannot be approximated well by a linear model. The bold red cross is the instance being explained, and the dashed line is the learned explanation that is locally faithful (from [1], Figure 3)

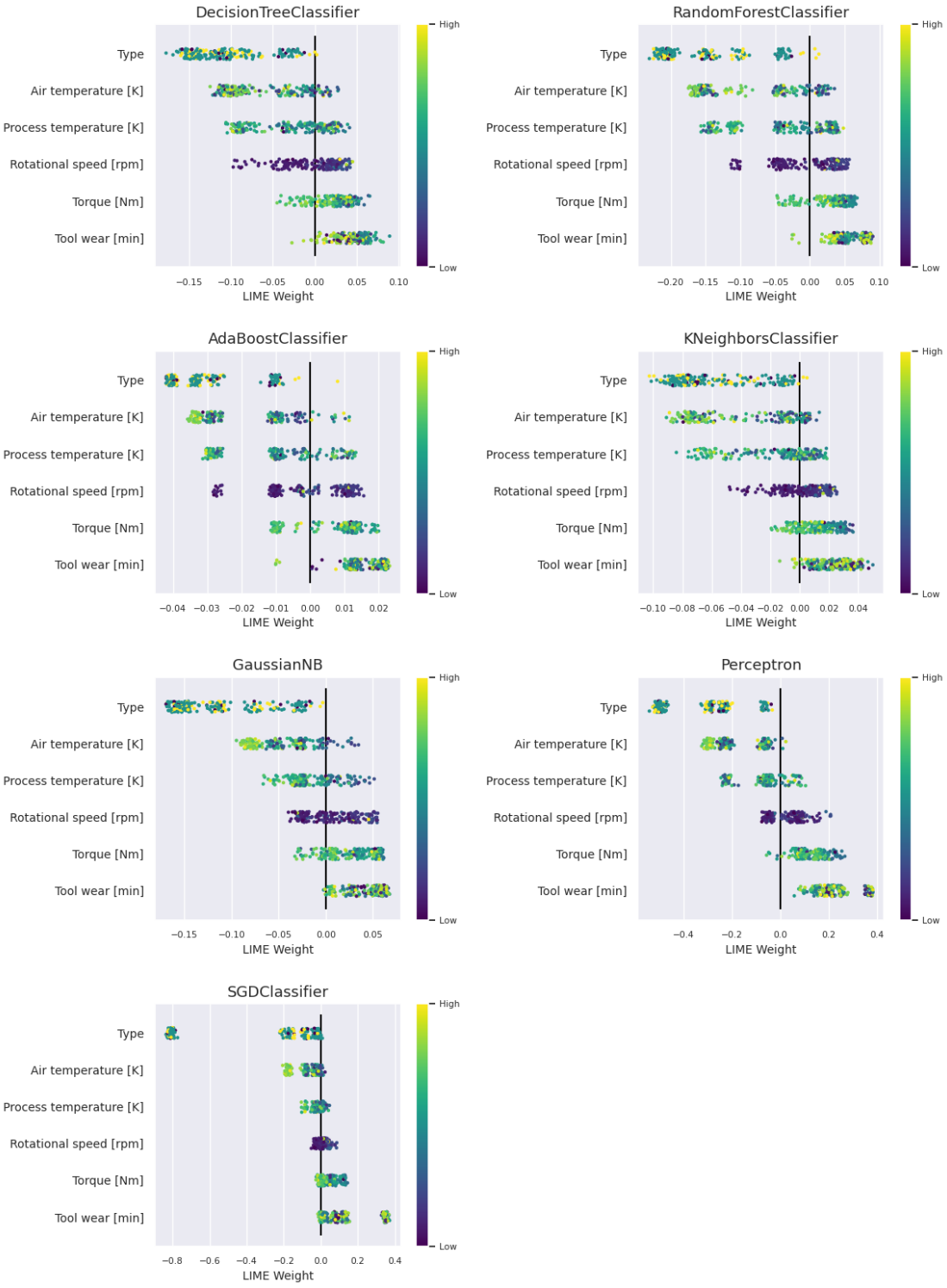


Figure 3: LIME summary

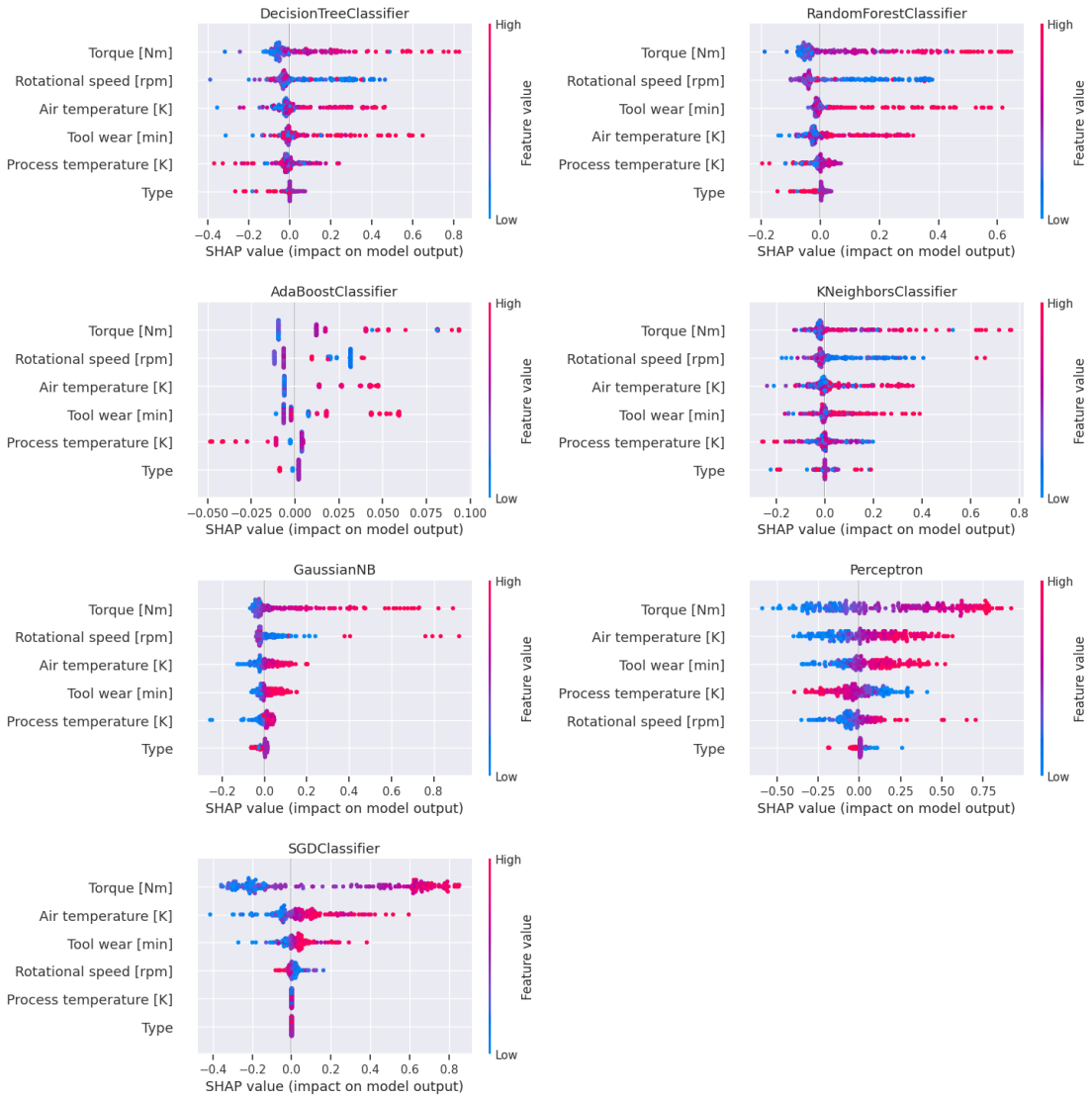
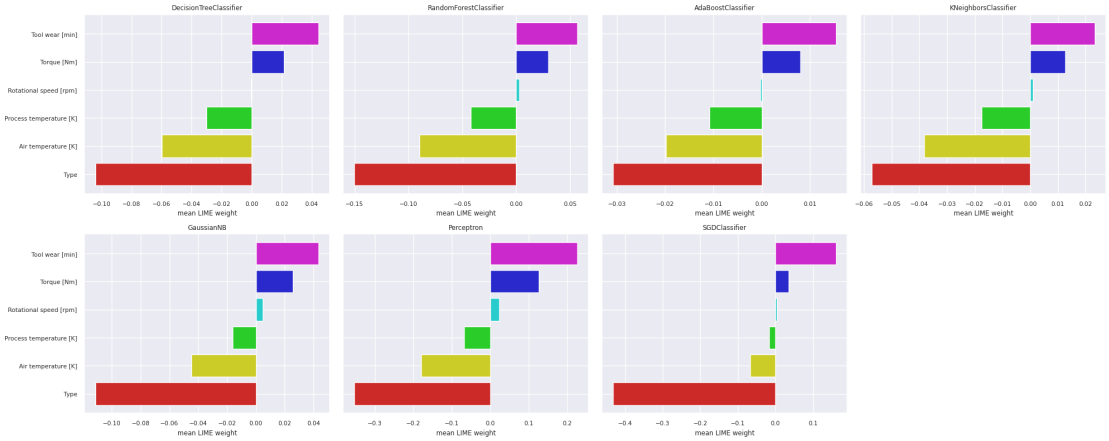
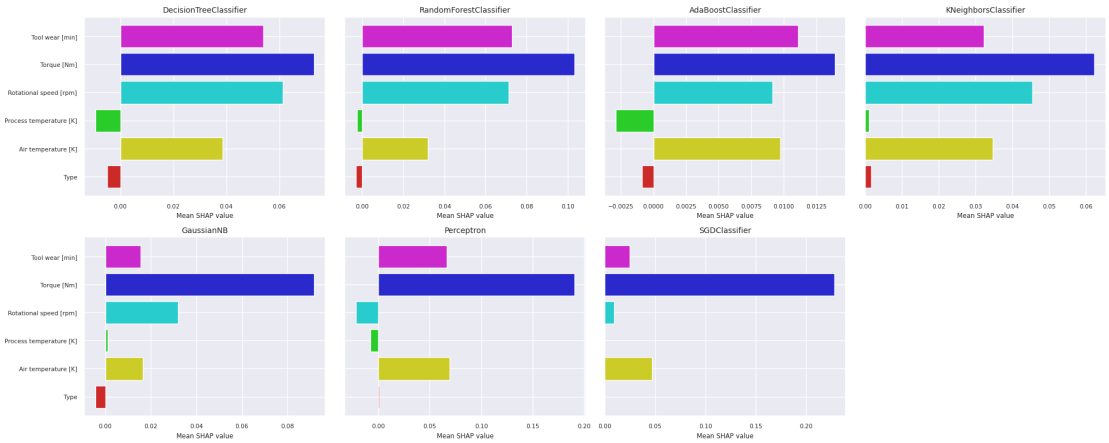


Figure 4: SHAP summary

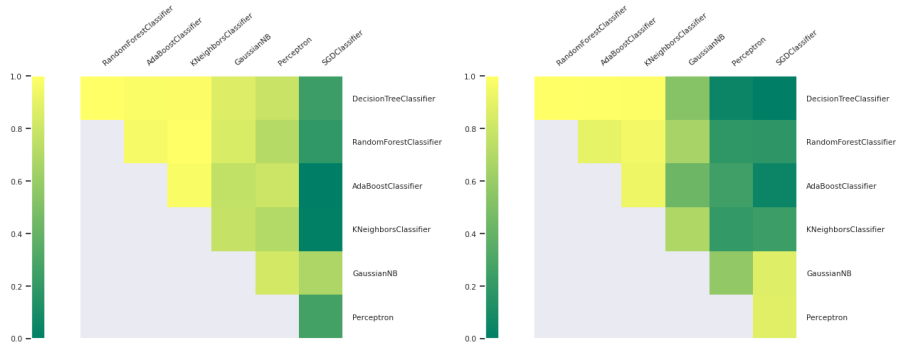


(a) LIME



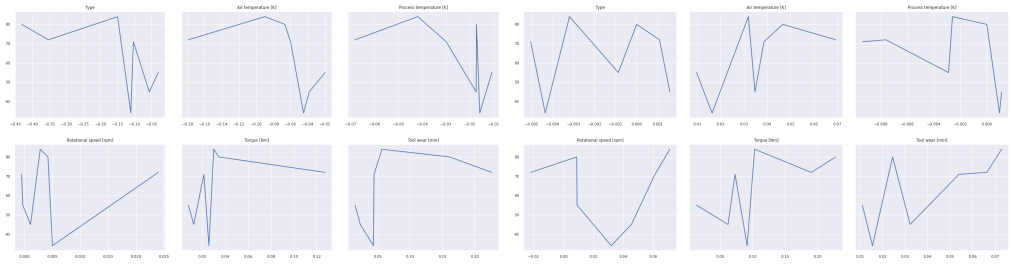
(b) SHAP

Figure 5: Similarity between mean absolute feature weights



(a) LIME **(b)** SHAP

Figure 6: Similarity between mean absolute feature weights



(a) LIME **(b)** SHAP

Figure 7: Model performance (average class 1 recall) vs feature weight

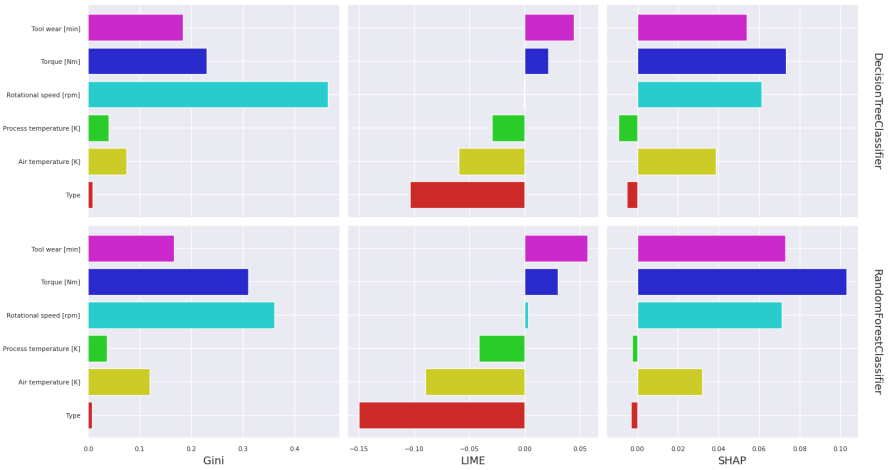


Figure 8: Gini vs LIME vs SHAP importances for tree based models