Chapter 12
Global Study: Influence of Tuning

Martin Zaefferer, Olaf Mersmann, and Thomas Bartz-Beielstein

Abstract Expanding the more focused analyses from previous chapters, this chapter takes a broader view at the tuning process. That means, rather than tuning an individual model, this investigation considers the tuning of multiple models, with different tuners, and varying data sets. The core aim is to see how characteristics of the data and the model choice may impact the tuning procedure. We investigate five hypotheses, concerning the necessity of tuning, the impact of data characteristics, the impact of the target variable type, the impact of model choice, and benchmarking. Not only does this entail an in-depth tuning study, but we also tie our results to a measure of problem difficulty and use consensus ranking to aggregate the diverse experimental results.

12.1 Introduction

As described in Sect. 4.4, the Sequential Parameter Optimization Toolbox (SPOT) offers a robust approach for the tuning of Machine Learning (ML) algorithms, especially if the training and/or evaluation run time become large.

In practice, the learning process of models, $\mathcal{A}$, and hence the choice of their hyperparameters, λ, is influenced by a plethora of other factors. On the one hand, this complex situation further motivates the use of tuning procedures, since the ML algorithms have to be adapted to new data or situations. On the other hand, this

M. Zaefferer (✉)
Bartz & Bartz GmbH and with Institute for Data Science, Engineering, and Analytics, TH Köln, Gummersbach, Germany

Duale Hochschule Baden-Württemberg Ravensburg, Ravensburg, Germany
e-mail: zaefferer@dhbw-ravensburg.de

O. Mersmann · T. Bartz-Beielstein
Institute for Data Science, Engineering, and Analytics, TH Köln, Steinmüllerallee 1, 51643 Gummersbach, Germany
e-mail: olaf.mersmann@th-koeln.de

T. Bartz-Beielstein
e-mail: thomas.bartz-beielstein@th-koeln.de

E. Bartz et al. (eds.), *Hyperparameter Tuning for Machine and Deep Learning with R*,
https://doi.org/10.1007/978-981-19-5170-1_12

raises the question of how such factors influence the tuning process itself. We want to investigate this in a structured manner.

In detail, the following factors, which are introduced in Sect. 8.2.1, are objects of our investigation.[1] We test their influence on the tuning procedures.

- Number of numerical features in the data (`nnumericals`).
- Number of categorical features in the data (`nfactors`).
- Cardinality of the categorical features, i.e., the maximal number of levels (`cardinality`).
- Number of observations in the data (n).
- Task type, classification or regression.
- Choice of model.

12.2 Research Questions

We want to investigate the following hypotheses.

(H-1) Tuning is necessary to find good parameter values (compared to defaults).
(H-2) Data: Properties of the data influence (the difficulty of) the tuning process.

- Information content: If the data has little information content, models are easier to tune, since more parameter configurations achieve near-optimal quality. In general, changing parameters has less impact on model quality in this case.
- Number of features: A larger number of features leads to longer run times, which affects how many evaluations can be made during tuning.
- Type of features: The number of numerical and/or categorical features and their cardinality influences how much information is available to the model, hence may affect the difficulty of tuning.
- Number of observations n: With increasing n, the average run time of model evaluations will increase.

(H-3) Target variable: There is no fundamental difference between tuning regression or classification models.
(H-4) Model: The choice of model (e.g., Elastic Net (EN) or Support Vector Machine (SVM)) affects the difficulty of the tuning task, but not necessarily the choice of tuning procedure.
(H-5) Benchmark: The performance of the employed tuners can be measured in a statistically sound manner.

[1] The data from Census-Income (KDD) Data Set (CID) is historical. It includes wording or categories regarding people which do not represent or reflect any views of the authors and editors.

12.3 Setup

To investigate our research questions, the data set needs to be pre-processed accordingly. This pre-processing is documented in the included source code and is explained in Sect. 8.2.1.

For classification, the experiments summarized in Table 12.1 are performed with each tuner and each model. A reduced number of experiments are performed for regression, see Table 12.2.

To judge the impact of creating random subsets of the data (i.e., to reduce n), and to consider the test/train split, three data sets are generated for each configuration. All experiments are repeated for each of those three data sets.

Table 12.1 Experiments for classification: Investigated combinations of the number of categorical features (nfactors), numerical features(nnumericals), cardinality, and n. An empty field for cardinality occurs for low nfactors. In that case, no categorical features are present, so the number of categories becomes irrelevant. The number of observations, n, is varied on a logarithmic scale with five levels in the range from 10^4 to 10^5, i.e., 10 000, 17 783, 31 623, 56 234, and 100 000

nfactors	nnumericals	Cardinality	Number of observations (n)
High	Low	Low	$10^4, 10^{4.25}, \ldots, 10^5$
Medium	Medium	Low	$10^4, 10^{4.25}, \ldots, 10^5$
Low	High		$10^4, 10^{4.25}, \ldots, 10^5$
High	High	Low	$10^4, 10^{4.25}, \ldots, 10^5$
High	Low	Medium	$10^4, 10^{4.25}, \ldots, 10^5$
Medium	Medium	Medium	$10^4, 10^{4.25}, \ldots, 10^5$
High	High	Medium	$10^4, 10^{4.25}, \ldots, 10^5$
High	Low	High	$10^4, 10^{4.25}, \ldots, 10^5$
Medium	Medium	High	$10^4, 10^{4.25}, \ldots, 10^5$
High	High	High	$10^4, 10^{4.25}, \ldots, 10^5$
Complete data set			299285

Table 12.2 Experiments for regression: Investigated combinations of the number of categorical features (nfactors), numerical features (nnumericals), cardinality, and n. An empty field for cardinality occurs for low nfactors. In that case, no categorical features are present, so the number of categories becomes irrelevant

nfactors	nnumericals	Cardinality	Number of observations (n)
low	High		$10^4, 10^{4.25}, \ldots, 10^5$
High	High	Low	$10^4, 10^{4.25}, \ldots, 10^5$
High	High	High	$10^4, 10^{4.25}, \ldots, 10^5$

12.3.1 Model Configuration

The experiments include the models k-Nearest-Neighbor (KNN), Decision Tree (DT), EN, Random Forest (RF), Gradient Boosting (GB), and SVM. Their respective parameters are listed in Table 4.1.

For EN, `lambda` is not optimized by the tuner. Rather, the `glmnet` implementation itself tunes that parameter. Here, a sequence of different `lambda` values is tested (Hastie and Qian 2016; Friedman et al. 2020).

For SVM, the choice of the kernel (`kernel`) is limited to `radial` and `sigmoid`, since we experienced surprisingly large runtimes for `linear` and `polynomial` in a set of preliminary experiments. Hence, `degree` is also excluded, as it is only relevant for the `polynomial` kernel. Due to experiment runtime, we also did not perform experiments with SVM and KNN on data sets with $m \geq 10^5$ observations. These would require using model variants that are able to deal with huge data sets (e.g., some sparse SVM type).

Table 3.8 lists hyperparameters that are actually tuned in the experiments, including data type, bounds, and employed transformations. Here, we mostly follow the bounds and transformations as used by Probst et al. (2019a). Fundamentally, these are not general suggestions. Rather, reasonable bounds will usually require some considerations with respect to data understanding, modeling/analysis, and computational resources. Bounds on values which affect run time should be chosen so that experiments are still possible within a reasonable time frame. Similar consideration can apply to memory requirements. Where increasing/decreasing parameters may lead to increasing/decreasing sensitivity of the model, a suitable transformation (e.g., log-transformation) should be applied.

Most other configurations of the investigated models remain at default values. The only exceptions are:

- `ranger`: For the sake of comparability with other models, model training and evaluation are performed in a single thread, without parallelization (`num.threads = 1`).
- `xgboost`: Similarly to `ranger`, we set `nthread=1`. For regression, the evaluation metric is set to the root-mean-square error (`eval_metric="rmse"`). For classification, log-loss is chosen (`eval_metric="logloss"`).

12.3.2 Runtime For the Global Study

Similar to the local studies (Chaps. 8–10), run times are recorded in addition to Mean Mis-Classification Error (MMCE) (MMCE is defined in Eq. (2.2)). The recorded run times are overall run time of a model evaluation, run time for prediction, and run time for training.

Runtime budget: To mirror a realistic use case, we specify a fixed run time budget for the tuner. This limits how long the tuner may take to find potentially optimal

hyperparameter values. We set a budget of 5 h for SVM, KNN, RF, and GB. Since EN and DT are much faster to evaluate and less complex, they receive a considerably lower budget (EN: 1 h, DT: 5 min).

Timeout: For a majority of the models, the run time of a single evaluation (training + prediction) is hard to predict and may easily become excessive if parameters are chosen poorly. In extreme cases, the run time of a single evaluation may become so large that it consumes the bulk of the tuner's allotted run time, or more. In such a case, there would be insufficient time to test different hyperparameter values. To prevent this, we specify a limit for the run time of a single evaluation, which we call timeout. If the timeout is exceeded by the model, the evaluation will be aborted. During the experiments, we set the timeout to a twentieth of the tuner's overall run time budget. Exceptions are the experiments with DT (`rpart`): Since `rpart` evaluates extremely quickly, (in our experiments: usually much less than a second) and has a correspondingly reduced run time budget (5 min), the timeout is not required. In fact, tracking the timeout would add considerable overhead to the evaluation time in this case.

Additional performance measure from the `mlr` package could be easily integrated when necessary.[2]

12.3.2.1 Surrogate Model

In one important case, we deviate from more common configurations of surrogate model based optimization algorithms: For the determination of the next candidate solution to be evaluated, we directly use the predicted value of the Gaussian process model, instead of the so-called expected improvement. Our reason is, that the expected improvement may yield worse results if the number of evaluations is low, or the dimensionality rather high (Rehbach et al. 2020; Wessing and Preuss 2017). With the strictly limited run time budget, our experience is that the predicted value is preferable. A similar observation is made by De Ath et al. (2019).

12.3.2.2 Configuration of Random Search

With Random Search (RS), hyperparameter values will be sampled uniformly from the search space. All related configurations (timeout, run time budget, etc.) correspond to those of SPOT.

[2] A complete list of measures in `mlr` can be found at https://mlr.mlr-org.com/articles/tutorial/measures.html.

12.3.3 Benchmark of Runtime

The experiments are not run on entirely identical hardware, but on somewhat diverse nodes of a cluster. Hence, we have to consider that single nodes are faster or slower than others (i.e., a tuning run on one node may perform more model evaluations than on another node, simply because of having more computational power). To preserve some sort of comparability, we compute a corrective factor that will be multiplied with the run time budget. Before each tuning run, we compute a short performance benchmark. The time measured for that benchmark will be divided by the respective value measured at a reference node, to determine the run time multiplicator. We use the `benchmark_std` function from the `benchmarkme` R package (Gillespie 2021).

12.3.4 Defaults and Replications

As a comparison basis, we perform an additional experiment for each model, where all hyperparameter values remain at the models default settings. However, in those cases we do *not* set a timeout for evaluation. Since no search takes place, the overall run time for default values is anyways considerably lower than the run time of SPOT or RS. All other settings correspond to those of SPOT and RS. To roughly estimate the variance of results, we repeat all experiments (each tuner for each model on each data set) three times.

12.4 Results

In this section, we provide an exploratory overview of the results of the experiments. A detailed discussion of the results in terms of the research questions defined in Sect. 12.2 follows in Sect. 12.5.

To visualize the overall results, we show exemplary boxplots.[3] Since different results are achieved depending on the data set and optimized model, a preprocessing step is performed first: For each sampled data set and each model, the mean value of all observed results (model quality of the best solutions found) is determined. This mean is then subtracted from each individual observed value of the corresponding group. Subsequently, these subtracted individual values are examined. This allows a better visualization of the difference between the tuners without compromising interpretability. The resulting values are no longer on the original scale of the model quality, but the units remain unchanged. Thus, differences on this scale can still be interpreted well.

[3] Corresponding boxplots for all experiments can be found in the appendix of this document.

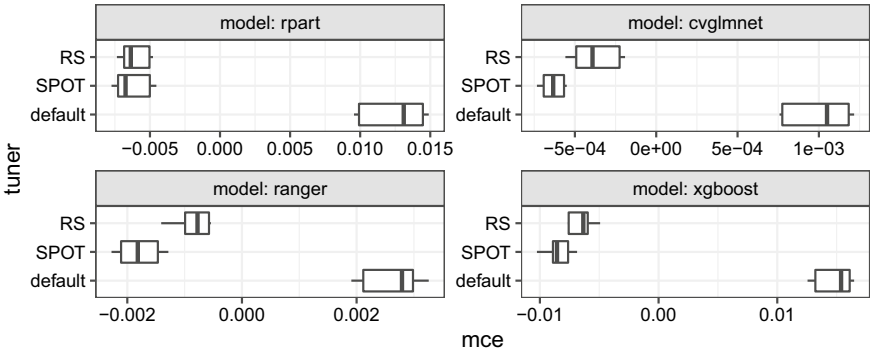

Fig. 12.1 Boxplot comparing tuners for different classification models, with $m = 10^5$, nfactors = high, nnumericals = high, cardinality = high. For this value of n, no experiments with KNN and SVM are performed. The presented quality measure (MMCE) shows values after preprocessing, where the mean value for each problem instance is subtracted

For the classification experiments, Fig. 12.1 shows the results for a case with many features and observations (nfactors, nnumericals, and cardinality are all set to high and $m = 10^5$). The figure first shows that both tuners (RS, SPOT) achieve a significant improvement over default values. The value of this improvement is in the range of about 1% MMCE. In almost all cases, the tuners show a smaller dispersion of the quality values than a model with default values. Except for the tuning of `rpart`, the results obtained by SPOT are better than those of RS.

Likewise for classification, Fig. 12.2 shows the case with $n = 10^4$, without categorical features (nfactors=low) and with maximum number of numerical features (nnumericals=high). Here the data set contains much less information, since a large part of the features is missing and only few observations are available. Still, both tuners are significantly better than using default values. In this case, it is mostly not clear which of the two tuners provides better results. For `rpart` (DT) and SVM, RS seems to work better.

In the same form as for classification, Figs. 12.3, and 12.4 show results for regression models. Unlike for classification, the results here are somewhat more diverse. In particular, `glmnet` shows a small difference between the tuners and default values. There are also differences of several orders of magnitude between the individual models (e.g. RF and SVM). For example, for RF the differences between tuners and default values are about 0.02 years (the target variable scale is age in years). As shown in Figs. 12.3 and 12.4, the interquartile range is about 0.01 years.

For GB, on the other hand, there is a difference of about 20 years between tuners and default values. Here, the default values seem to be particularly poorly suited.

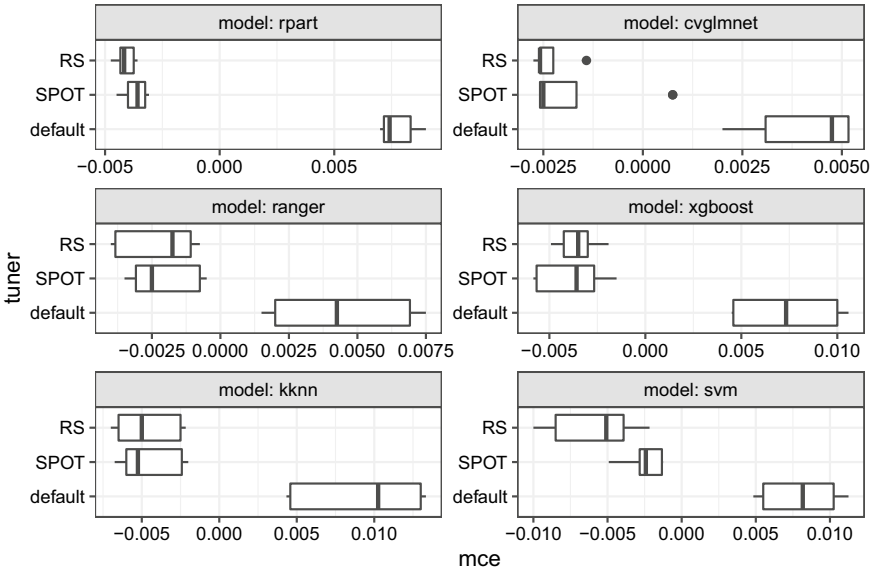

Fig. 12.2 Boxplot comparing tuners for different classification models, with $m = 10^4$, nfactors = low, nnumericals = high. The presented quality measure (MMCE) shows values after preprocessing, where the mean value for each problem instance is subtracted

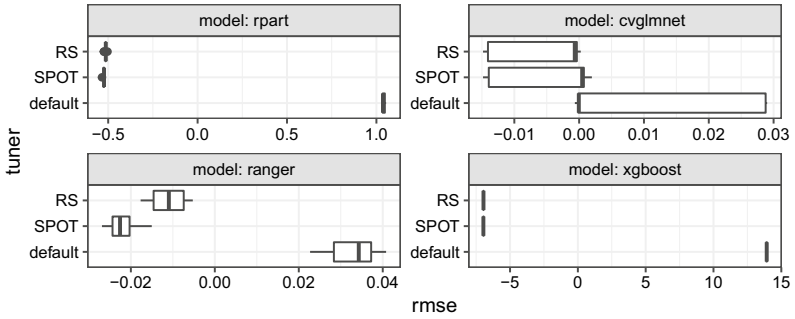

Fig. 12.3 Boxplot comparing tuners for different regression models, with $m = 10^5$, nfactors = high, nnumericals = high, cardinality = high. For this value of n no experiments with KNN and SVM are performed. The presented quality measure (MMCE) shows values after preprocessing, where the mean value for each problem instance is subtracted

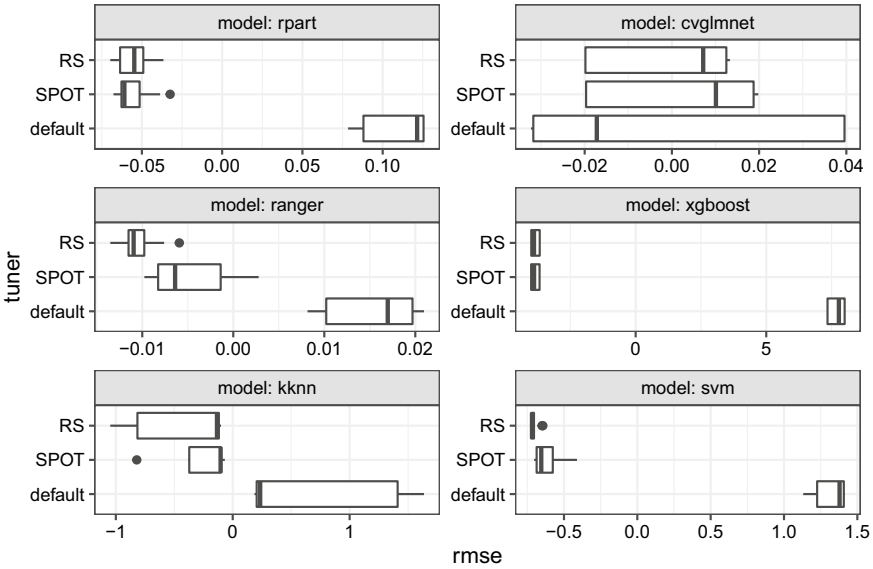

Fig. 12.4 Boxplot comparing tuners for different regression models, with $m = 10^4$, nfactors = low, nnumericals = high. The presented quality measure (MMCE) shows values after preprocessing, where the mean value for each problem instance is subtracted

12.5 Discussion

12.5.1 Rank-Analysis

We analyze the results of the experiments using rankings of the result values instead of the raw results. Rankings are scale invariant, so aggregating the results from different problem instances (resulting from data set and model choice) with different scales for the result values is possible. This is important because we have to aggregate over very diverse problem instances.

To aggregate rankings (also called consensus rankings), we follow the so-called "optimal ranking" approach of Kemeny (1959). Here, the consensus ranking is determined such that the mean distance between the consensus ranking and observed ranking is minimal. The distance measure used for this purpose is Kendall's tau (Kendall 1938), which counts the number of pairwise comparisons in which two rankings contradict each other. The ranking determined using this approach can be interpreted as the "median" of the individual rankings.

This procedure has the following advantages (Hornik and Meyer 2007; Mersmann et al. 2010a):

- Scale invariant.
- Invariant to irrelevant alternatives.

- Aggregation of large sets of comparisons, over arbitrary factors.
- Easy/intuitive to interpret results and visualizable.
- Generates relevant information: selection of the best one.
- Additional weights for preferences can be inserted.
- Fast evaluation.
- Non-parametric method, no distribution assumptions.
- Visualization of clusters over distance is possible, identification of problem classes with similar algorithm behavior.

However, in addition to these advantages, there are also disadvantages:

- Estimating uncertainty in the ranking is difficult.
- The ranking does not have to induce a strict ordering, ties are possible.

We generate the consensus ranking by combining rankings of tuners (SPOT, RS, default) of individual experiments. We always summarize the rankings of 9 experiments (3 repetitions of the tuner runs on each of 3 randomly drawn data sets). Then, to aggregate across different variables related to the study subjects (e.g. n, nfactors), we count how often the tuners reach a certain consensus rank.

This count is divided by the total number of experiments (or number of rankings) on the corresponding problem instances. Thus, we record for each tuner how often a particular consensus rank is achieved.

Simplified example:

- For case 2 (i.e., for a fixed choice of nnumericals, nfactors, cardinality, model, n, target variable), the comparison of SPOT, RS, and default methods resulted in the ranks

$$\{1, 3, 2\} \quad \{1, 2, 3\} \quad \{2, 1, 3\}.$$

  The consensus ranking for this case is $\{1, 2, 3\}$.
- For case 2 (i.e., for *another* fixed choice of nnumericals, nfactors, cardinality, model, n, target variable), the comparison of SPOT, RS, and default methods resulted in the ranks

$$\{3, 2, 1\} \quad \{1, 2, 3\} \quad \{2, 1, 3\}.$$

  The consensus ranking for this case is $\{2, 1, 3\}$.
- When both experiments are combined for an analysis, the frequencies for the obtained rankings are as follows:
  - SPOT: rank 1 with 50%, rank 2 with 50%, rank 3 with 0%.
  - RS: rank 1 with 50%, rank 2 with 50%, rank 3 with 0%.
  - Default: rank 1 with 0%, rank 2 with 0%, rank 3 with 100%.

12.5.2 Rank-Analysis: Classification

Based on the analysis method described in Sect. 12.5.1, Fig. 12.5 shows the relationship between the tuners, the number of observations n, and the optimized models. It shows that SPOT and RS mostly beat the default setting and SPOT also usually performs better than RS. However, some cases deviate from this. Especially, `glmnet` (EN) and `rpart` (DT) seem to profit less from tuning: here the distinction between the ranks of tuners is more difficult. In addition, when the number of observations is small, there tends to be a greater uncertainty in the results. These results can be partly explained by the required runtime of the models. With a smaller number of observations, the runtime of the individual models decreases, and `glmnet` and `rpart` are the models with the lowest runtime (in the range of a few seconds, or below one second in the case of `rpart`). If the evaluation of the models itself takes hardly any time, it is advantageous for RS that its runtime overhead is low. SPOT, on the other hand, requires a larger overhead (for the surrogate model and the corresponding search for new parameter configurations).

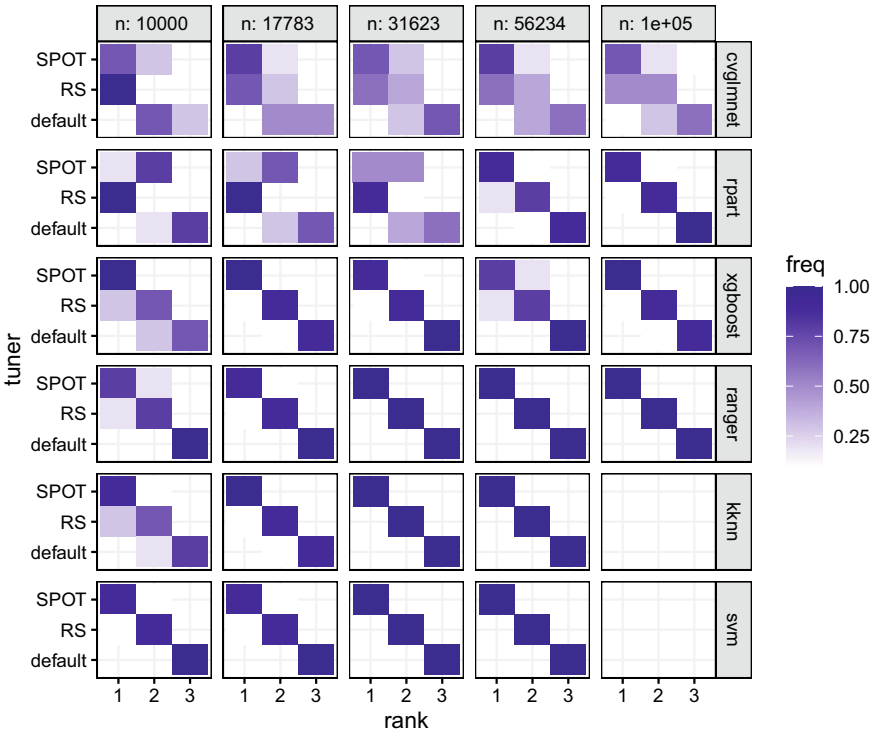

Fig. 12.5 Rank of tuners depending on number of observations (n) and model, for classification

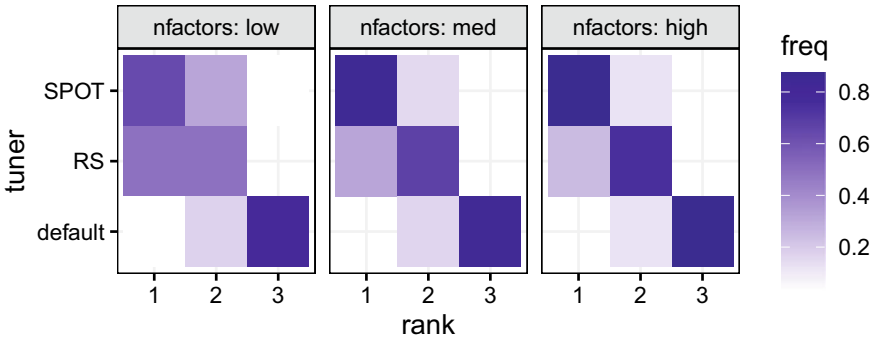

Fig. 12.6 Rank of tuners as a function of the number of categorical features (nfactors) and the model, for classification

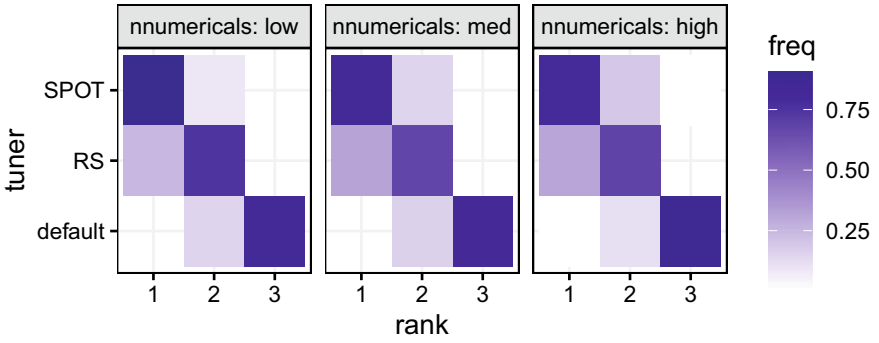

Fig. 12.7 Rank of tuners depending on number of numerical features (nnumericals) and model, for classification

Figure 12.6 shows the corresponding summary of the results depending on the number of categorical features (nfactors). Again, SPOT usually performs best, followed by RS. There is a tendency for the greatest uncertainty to be found in cases where number of categorical features is low. Two explanations are possible: On the one hand, the reduction of features also means a reduction of the required runtime. On the other hand, the difficulty of the modeling increases, since with fewer features less information is available to separate the classes.

The corresponding results for the number of numerical features can be found in Fig. 12.7. There are hardly any differences, the number of numerical features seems to have little influence. It should be noted that the data set contains fewer numerical than categorical features anyway.

The cardinality of the categorical features also has little influence, see Fig. 12.8. However, there is a slight tendency: at higher cardinality, the distinction between the first rank (SPOT) and second rank (RS) is clearer. This can be explained (similarly to nfactors) by the higher information content of the data set.

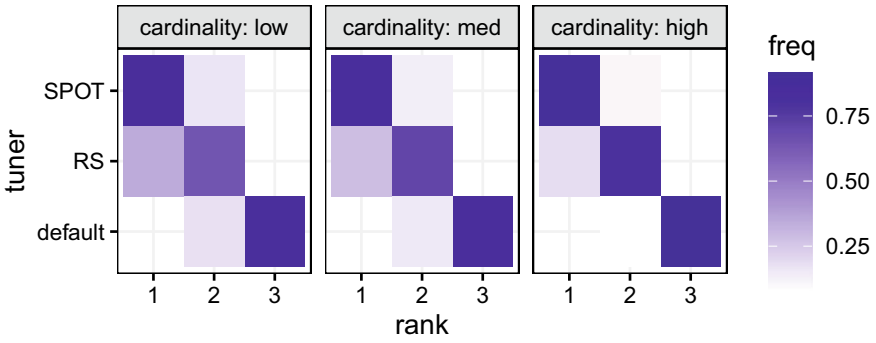

Fig. 12.8 Rank of tuners as a function of the cardinality of categorical features and the model, for classification. *Note* This figure does not include the cases where the data set no longer contains categorical features (cardinality cannot be determined)

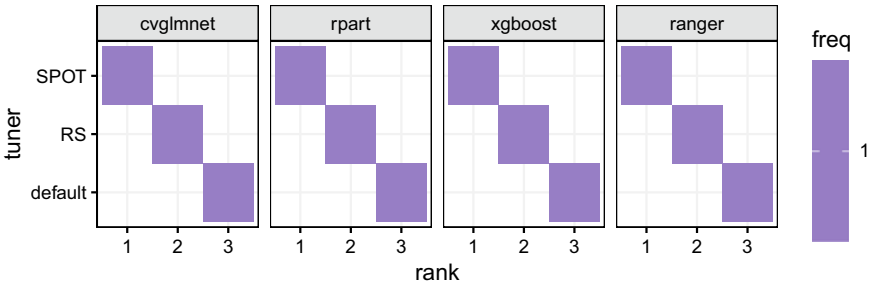

Fig. 12.9 Rank of tuners depending on the model, for classification on the complete data set. Due to the increased runtime, `kknn` (KNN) and `e1071` (SVM) are not included

Finally, Fig. 12.9 shows the result of the tuners on the unmodified, complete data set. For each case, SPOT gets rank 1 and RS rank 2. This result is in line with the trends described above, since the complete data set contains the most information and also leads to the largest runtimes (for model evaluations).

12.5.3 Rank-Analysis: Regression

In addition to the classification experiments, a smaller set of experiments with regression as an objective are also conducted. Figure 12.10 shows the results for this case separately for each optimized model. Here, too, SPOT is usually ranked first. Unlike in case of classification, however, there is more uncertainty.

For `glmnet` (EN), default values occasionally even achieve rank 1. It seems that linear modeling with `glmnet` is unsuitable for regression with the present data set,

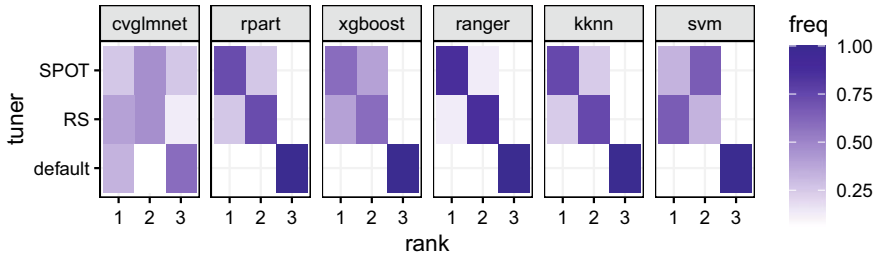

Fig. 12.10 Rank of tuners depending on model for regression

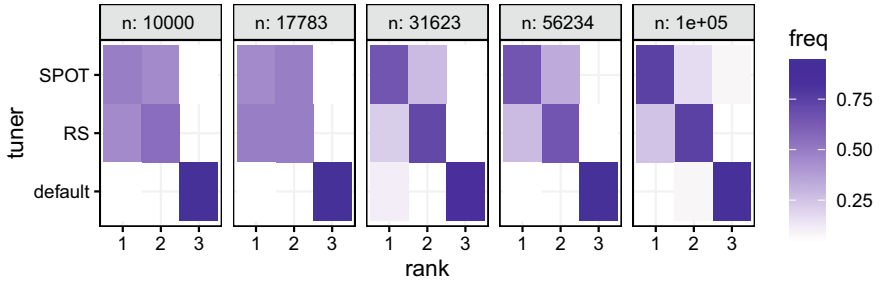

Fig. 12.11 Rank of tuners, depending on number of observations (n), for regression

so that the tuners can hardly achieve differences to the default values. This behavior was already indicated in Fig. 12.4.

In the case of SVM, RS is more often ranked first than SPOT. The reason for this is not clear. A possible cause is a lower influence of the hyperparameters on the model quality of SVM for regression (compared to classification). However, it should also be considered that less experiments were conducted for regression than for classification.

The dependence on the number of observations n is shown in Fig. 12.11. The results for regression with respect to n are largely consistent with those for classification. With an increasing number of observations, SPOT is more clearly ahead of RS.

The correlation with categorical features (number, cardinality) is also consistent with the classification results, see Fig. 12.12. With larger number of features and larger cardinality, a clearer separation between the tuners is observed.

12.5.4 Problem Analysis: Difficulty

In the context of this study, an interesting question arose as to how the difficulty of the modeling problem is related to the results of the tuning procedures. We investigate this on the basis of the data obtained from the experiments.

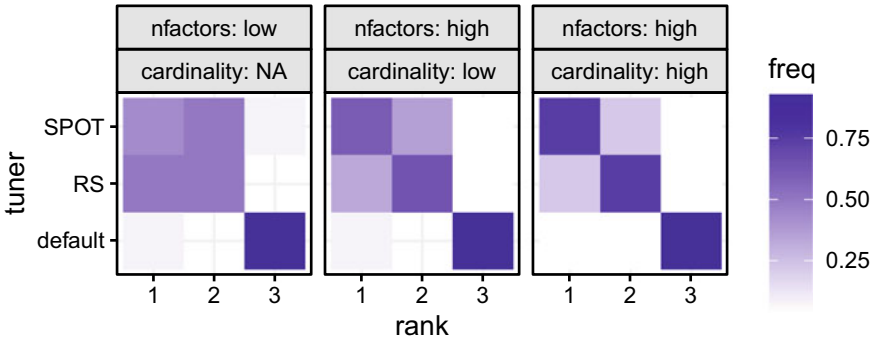

Fig. 12.12 Rank of tuners, depending on cardinality and number of categorical features (nfactors), for regression

In general, there are many measures of the difficulty of modeling problems in the literature. An overview is given by Lorena et al. (2019). Of these measures, the overlap volume (F2, see Lorena et al. 2019) is of interest, since it is easily interpretable and not specific to a particular model.

First, the values of each feature are separated into two groups, one for each class. Then, the range of the intersection of the two groups is computed for each individual feature. This is called the *overlap* of a feature. The *overlap volume* of the data set is then the product of the individual overlap values.

However, this measure is unsuitable for categorical features (even after dummy coding). Furthermore, outliers are very problematic for this measure.

We therefore use a slight modification: We calculate the proportion of sample values for each feature, which could occur in both classes (i.e. for which a swap of the classes based on the feature value is possible). As an example, this is illustrated for a numeric and a categorical feature in Fig. 12.13. For the overall data set, the individual overlap values of each feature are multiplied. Subsequently, we refer to this measure as *sample overlap*.

Figure 12.14 shows the dependence of the sample overlap on our data properties (n, nfactors, nnumericals, cardinality).

Our data sets can be grouped into 4 difficulty levels based on these values of sample overlap:

1. Sample overlap ≈ 0.39: nfactors = high and cardinality = high (green).
2. Sample overlap ≈ 0.54: nfactors = high and cardinality = med (blue).
3. Sample overlap ≈ 0.76: all others (orange).
4. Sample overlap ≈ 1.00: nfactors = low (red).

Here, 4 corresponds to the highest level of difficulty. For the range relevant in the experiments, there is almost no change depending on n or nnumericals. For nfactors and cardinality a strong correlation can be seen.

Based on the 4 difficulty levels, the ranks already determined in previous sections can be re-ranked. The result is summarized in Fig. 12.15. It turns out that as the

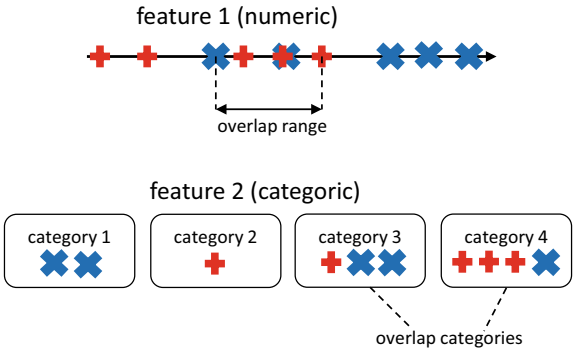

Fig. 12.13 Example of the sample overlap for two features of a classification problem with two classes. The crosses show samples from the data set. All samples with class 1 are red (+). All samples with class 2 are blue (x). For feature 1, the overlap is 50% (number of samples in the overlap area divided by the total number of samples). For feature 2, the overlap is 70% (number of samples in the overlap categories divided by total number of samples). For both characteristics together, the sample overlap is $0.5 \times 0.7 = 0.35$

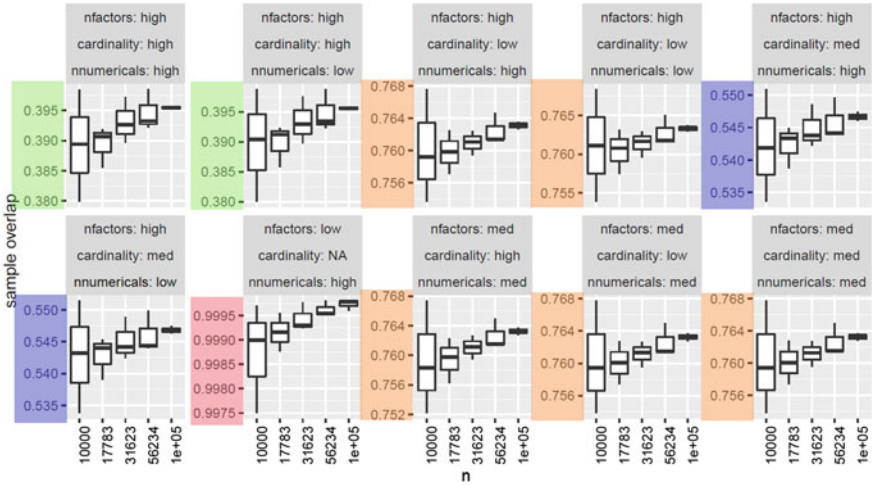

Fig. 12.14 Sample overlap depending on the properties of the data set varied in the experiments. The sample overlap is used as a measure of problem difficulty and leads to the definition of four difficulty levels (marked in color)

Fig. 12.15 Frequency of the achieved ranks of the tuners depending on the difficulty level

difficulty of the problem increases, the rank differences are less robust. That is, with larger sample overlap it is harder to estimate which tuner works best. This is plausible with respect to the theoretical extreme case: A maximally difficult data set cannot be learned by any model, since the features no longer contain any information about the classes. In this case, the hyperparameters no longer have any influence and the tuners cannot improve.

12.6 Summary and Discussion

To summarize our findings, we discuss the hypotheses from Sect. 12.2.

(H-1) *Tuning is necessary:* In almost all cases, significantly worse results are obtained with default settings. In addition, the variance of the results decreases when tuners are used (see Figs. 12.1, 12.2, 12.3 and 12.4).

(H-2) *Data:* Differences between tuners become apparent for data sets with a high information content and for larger data volumes. If the number of levels or features (categorical and numerical) decreases, it can be observed that the differences between the tuners become smaller.
It has been confirmed that as the number of features and observations increases, the runtime increases. As the mean runtime of the models increases (i.e., for more complex models or larger data sets), SPOT performs increasingly better than RS, as the ratio of overhead to evaluation time decreases more for SPOT.

(H-3) *Target variable:* The execution of the tuning is not affected by the change of the target variable. A peculiarity that requires further investigation occurred when tuning SVM for regression: RS seems to perform better than SPOT. We recommend using a larger data base to investigate this case.

(H-4) *Model:* The choice of tuning method is not fundamentally influenced by the models. Models that can be evaluated very quickly (e.g. `rpart`) benefit from the larger number of evaluations. This is due to the fact that time was chosen as a termination criterion.

(H-5) *Benchmark:* As described in Sect. 12.5.1, analysis methods based on consensus ranking can be used to evaluate the suitability of tuners in a simple and statistically valid way.

Overall, the result of this experimental investigation underlines that tuning of machine learning models is essential to produce good configurations of the hyperparameters and thus models of high quality.

It is especially important to use tuners at all, since the biggest differences are usually observed in comparison to default settings. However, there are also differences between tuners, which in our investigation are mostly favoring the use of SPOT, i.e. a model-based optimizers. For the prospective focus on tuning with relatively large data sets (large n and k), this is all the more evident, since the resulting high model runtime favors the use of model-based methods.

Since the number of hyperparameters is manageable ($<< 100$), the addition of a few more parameters does not significantly increase the complexity of the tuning. For both Random Search (Bergstra and Bengio 2012) and SPOT, the addition of a few parameters is harmless, even if they have only a small effect on the model quality.

Moreover, the analysis results in additional benefits:

- Possible software bugs can be detected,
- seemingly unimportant parameters are detected as important,
- unknown interactions can be uncovered.

The surrogate model allows to learn the influence of the hyperparameters on the model values. Unimportant parameters are consequently weighted less in the search.

Finally, we recommend for the selection of the tuner, in the context of this study:

- Random Search can be used when either very little time (order of magnitude: time is sufficient for a single-digit number of sequential evaluations) but a lot of parallel computing capacity is available or when models can be evaluated extremely fast (in the range of seconds).
- If time-consuming computations with complex data and models need to be performed in more time (with relatively limited parallel computing capacity), we recommend model-based tuning with SPO.
- In exceptional cases, which deviate from our considered objective of tuning with complex data and models (extremely large amount of computing time available, average evaluation times for the models), the use of surrogate model-free tuning methods may also be considered.