Comprehensive and empirical evaluation of machine learning algorithms for LC retention time prediction

Robbin Bouwmeester, Lennart Martens, and Sven Degroeve

VIB-UGent Center for Medical Biotechnology, VIB, Ghent, Belgium

Department of Biochemistry, Ghent University, Ghent, Belgium

February 2, 2018

Abstract

Liquid chromatography retention time prediction is rarely optimized in terms of features and machine learning algorithm. Even though performance of the trained model can greatly influence the results when used in targeted and non-targeted mass spectrometry experiments. A total of seven machine learning algorithms are evaluated on 36 distinct datasets. The performance of the learning algorithms are compared on different number of training instances, features and overall performance. We show that no learning algorithm performs the best overall. However, gradient boosting generally outperforms the other algorithms if there are sufficient training examples. Predictions from a Bayesian Ridge, Lasso or SVR model are different from the tree boosting algorithms, and blending different types of models decreases the error on outliers. In conclusion, decisions made in terms of features and the learning algorithm have a significant impact on the performance of a retention time prediction. The code used to generate the regression models, evaluate the models and produce the figures is available at:
https://github.com/RobbinBouwmeester/tRPredictionOverview.

1 Introduction

Mass spectrometry (MS) coupled to Liquid Chromatography (LC) is a popular technique for the high-throughput analysis of the metabolome and lipidome. LC separates the analytes based on their physicochemical properties [22, 3]. High performance LC pumps solvents (mobile phase) through a column that is filled with beads that have specific molecules covalently bound (stationary phase). The interaction of the analytes with the mobile and stationary phase determines the time a molecule is retained in a column, and is therefore called the retention time (t_R).

The acquisition time per MS spectrum limits the ability of the mass spectrometer to detect lower abundant analytes [34]. Coupling LC with MS enables longer run times of a sample and increases the chance of detecting low abundant analytes through seperation. In addition to an increased sensitivity, LC-coupling provides information about the physicochemical properties of the analyte that is complementary to the mass over charge measurement of the MS.

The retention time for the majority of metabolites is unknown or only known for a specific experimental setup, and as a result it cannot be used as a measure for identification of metabolites. Regression models attempt to fill the gap between experimentally measured retention times and known metabolite structures without known retention times. These models have previously been applied in targeted and untargeted MS experiments of lipids [1], metabolites [34, 10, 33] and peptides [19, 28, 15, 25].

Retention time prediction can be applied in targeted MS experiments to reduce the number of experiments needed to target the molecules of interest. In non-targeted MS predicted t_R have been used to differentiate between isobaric lipids [1]. The predicted retention times were used to halve the search space of candidate lipid identifications, while retaining the majority of correct identifications. For small metabolites (¡ 400 Da) t_R prediction has been successfully applied to filter 40 % of the false identifications [10]. For peptides, predictions have been used to score spectrum matches and increase the number of confident

identifications by 12 % [30].

Retention time prediction can be a useful addition to an MS experiment, but due to different experimental setups the modeling procedure is not trivial [29, 4]. The differences in experimental setup significantly influence the retention times, and this results in non-transferable knowledge. As a result t_R prediction models are lab and setup specific. Models that calibrate between different setups have been published, but are only applicable to a limited set of metabolites or setups [4]. This means that researchers generally fit a model for every setup and go through a process where multiple modelling decisions have to be made.

The regression model used for t_R prediction is generally fitted using a machine learning algorithm, and there is a large variety of algorithms to choose from [11]. Choosing an algorithm often depends on historical experience of the researcher with particular machine learning algorithms. For t_R prediction the SVR model is the most popular option [27, 25, 19, 13, 18, 7, 20, 1]. Other types of machine learning algorithms that have been applied to t_R prediction models are linear regression, neural networks and random forest [34, 21, 6]. In addition a recent state-of-the-art algorithm called gradient boosting is evaluated in this manuscript as well. Generally the choice for an algorithm is not justified in a t_R prediction publication. Even though a comparison between multiple algorithms has been made a trivial task with the sci-kit learn (python) [26] or caret (R) [16] packages.

In literature there is no comprehensive overview of machine learning algorithms for t_R prediction for a variety of experimental setups. Even though, as we will show, the choice for one machine learning algorithm over the other can significantly influence the performance. Here, a comparison is presented between many different types of machine learning algorithms on a large variety of datasets with measured retention times for validated metabolite identifications.

2 Methods

2.1 Machine learning algorithms

In this research a diverse set of machine learning algorithms are evaluated for their ability to compute accurate t_R prediction models from relatively limited amounts of data. The algorithms produce a regression model that maps between molecular descriptors (features) used to predict the t_R (target). A comparison between seven implementations of supervised machine learning algorithms is presented. The algorithms were selected based on their fundamental differences in the prediction models they fit as well as their popularity within the LC and machine learning community.

Linear models assume the relation between the features and the target to be linear and as such are limited in what can be modeled but are typically much more robust towards overfitting the data. We evaluated three popular linear models that differ mainly in the way the linear model parameters are fitted to the data. They are Bayesian Ridge Regression (BRR) [14], Least Absolute Shrinkage and Selection Operator regression (LASSO) [31] and Linear Support Vector Regression (LSVR) [9].

In contrast, non-linear models are able to model more complex relationships between the features and the target at the cost of an increased risk of overfitting, especially on small datasets with many noisy features. Here we evaluated the Support Vector Regression (SVR) model with a Gaussian Radial Basis kernel Function (RBF) [9], a feedforward Artificial Neural Network (ANN) and three decision tree ensemble models that differ in the way the decision trees are constructed: through Adaptive Boosting (AB) [12], Gradient Boosting (GB) [23] or bagging in a Random Forest (RF) [5].

2.2 Datasets

The machine learning models were fitted on a wide range of publicly available LC-MS datasets obtained from MoNA (http://mona.fiehnlab.ucdavis.edu/), Pre-

dRet [29] and [1] that each contained measurements for at least 40 unique molecules. In total 36 datasets are used that were acquired at different labs and different experimental setups (Table S1). The datasets amounts to 8305 molecules of which 6759 molecules are unique. The datasets contain molecules with a low molecular weight (lowest: 59.07 Da) to high molecular weight (highest: 2406.65 Da) (Figure S1); ranging from acetamide to lipids. Molecules that are duplicated within a dataset were removed based on duplicated SMILES representation [32].

2.3 Molecular descriptors (features)

RDKIT [17] is used to convert SMILES to 196 features of the molecules (Table S2). A selection of 151 features are used for training; selection is based on a filter for standard deviation of a feature (> 0.01) and correlation between features ($R^2 < 0.96$).

In addition, a smaller subset of eleven features (used in [1]) was selected to evaluate the risk of overfitting. These eleven features include an estimate of the hydrophobicity using the octanol/water partition coefficient (MolLogP, slogp_VSA1 and slogp _VSA2), molar refractivity (SMR), estimated surface area (LabuteASA and TPSA), average molecular weight (AMW), polarizability based on molar refractivity (smr_VSA1 and smr_VSA2) and electrostatic interactions (peoe_VSA1 and peoe_VSA2).

2.4 Performance Evaluation

As t_R prediction is specific to an experimental condition the number of annotated molecules available for fitting (or training) a regression model (the trainset) is typically limited. Therefore we first investigate how different regression models perform for different trainset sizes by computing learning curves as follows.

Per dataset 160 unique molecules are selected at random. From these selected molecules trainsets with increasing sizes are sampled. The remaining

molecules in a dataset constitute the testset used to evaluate model performance. More concrete from the 160 selected molecules 20 molecules are sampled at random without replacement as the first trainset. Regression models are optimized and fitted on this trainset and the resulting models are evaluated on the testset. Next another 20 molecules are sampled at random without replacement from the remaining 140 selected molecules and added to the trainset. A model is again optimized on the (bigger) trainset and evaluated on the (same) testset. This procedure is repeated until the trainset contains 160 molecules.

In order to assess the generalization performance on a sufficiently large testset only datasets with at least 320 unique molecules are used in this experiment. The datasets that meet this requirement are Eawag_XBridgeC18, FEM_long, RIKEN, Stravs and Taguchi_12. To investigate the reproducibility of the obtained results each experiment is repeated 10 times with different random seeds.

To estimate the generalization performance further the algorithms are evaluated using 10-fold cross-validation (10CV) on all datasets using all unique molecules.

The generalization performance of a fitted regression model is evaluated using three different performance measures: Mean Absolute Error (MAE), Median Absolute Error (MedAE) and Pearson correlation (r). Let $\hat{y}$ be the prediction of the model and y the experimentally observed retention time for all molecules n in a dataset; then the MAE is calculated using the following equation:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{1}$$

The MAE gives a good indication of the performance, but is sensitive to outliers and for this reason the Median Absolute Error is used as the main metric:

$$MedAE = median(|y - \hat{y}|) \tag{2}$$

The MAE or MedianAE can be hard to compare between different classes of problems. Especially when the error depends on the range of the elution times.

Therefore, in addition to the previous two metrics the Pearson correlation is used:

$$r = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2}} \tag{3}$$

Even though the MAE, MedianAE and correlation can be calculated for different datasets a direct comparison between each of them is not possible. A prediction error of 10 seconds will have more impact when the run time is 129 seconds (dataset RIKEN) as compared to a run time of 4089 seconds (dataset Taguchi). In order to make the error on datasets comparable the errors were normalized by dividing by the retention time of the last detected analyte:

$$MedAE_{normalized} = \frac{MedAE(y, \hat{y})}{max(y)} \tag{4}$$

2.5 Hyperparameter optimization

Learning algorithms have hyperparameters that need to be set by the user before fitting the data and used to fit the model parameters. Here, the hyperparameters of the learning algorithms are optimized using values that are randomly drawn from a prespecified distribution (code listing S1). Random optimization is generally able to find the optimal parameters significantly faster than a grid search, because it is hypothesized that it samples more of the parameter space for the same amount of iterations [2]. Due to the number of datasets and implemented CV-procedure this faster method of hyperparameter optimization is required. For each machine learning algorithm a total of 100 randomly selected hyperparameter sets are evaluated using a 10-fold cross-validation. The hyperparameter set with the best mean absolute error using cross-validation is used for fitting the complete trainset.

2.6 Code and dataset

The code to generate the models implements the following libraries: scikit-learn V0.18.0 [26], Pandas V0.19.021 [24], RDKIT V2016.03.1 [17] and XGBoost V0.4 [8]. The code used to generate the regression models, make predictions and figures in this paper is available at: https://github.com/RobbinBouwmeester/tRPredictionOverview

3 Results

SVR models are one of the most frequently applied algorithms for LC t_R prediction [27, 25, 19, 13, 18, 7, 20, 1]. Researchers often default to one machine learning algorithm for t_R prediction without evaluating the performance of other algorithms. In this section we show how well different machine learning algorithms perform on different t_R prediction tasks.

3.1 Prediction performance versus trainset size

First, the ability of different learning algorithms to generalize trainsets with different sizes is investigated. Figure 1 shows learning curves for each of the regression models for the five datasets with the most molecules. As expected the learning curves show that all regression models benefit from larger trainsets. For the five datasets used in this experiment the largest generalization performance gain is typically observed when doubling the dataset size from 20 to 40 training examples. Still, most models show best performance when trained on the the largest sample size of 160 examples, with adaptive boosting being the exception. Even though for RIKEN this algorithm performs best for the smaller trainsets, it shows clear overfitting on the larger trainset sizes and some minor overfitting on the FEM_long, Stravs and Taguchi datasets.

Although no clear performance difference between the linear and non-linear prediction models can be observed from Figure 1, we do see a large performance increase for the FEM_long, RIKIN and Taguchi datasets for non-linear models. For the latter the GB algorithm clearly performs best overall, except for the

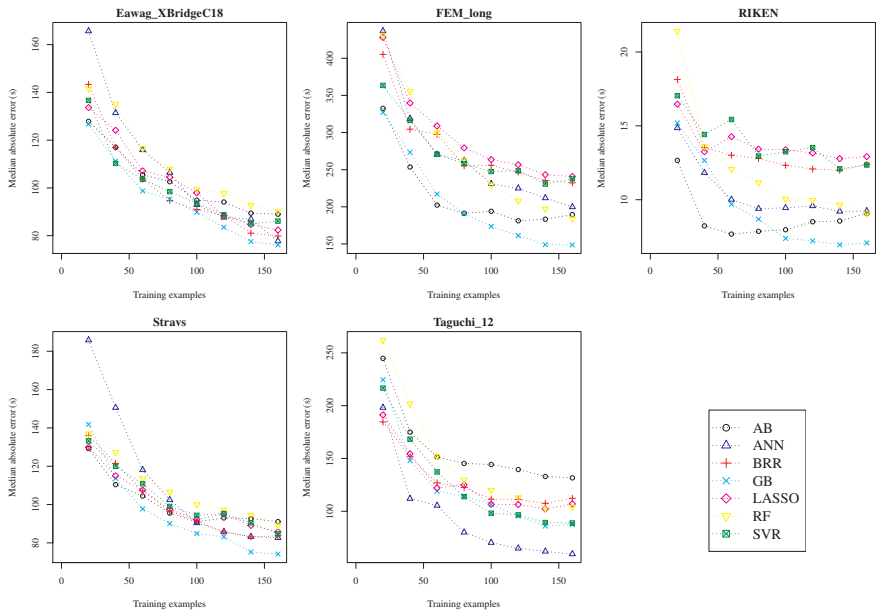

Figure 1: The learning curves for five datasets that have at least 320 training examples. The median absolute error (y-axis) is plotted against a specific number of training examples (x-axis). A total of 160 examples are selected for training and the remaining is used for testing. From the 160 examples a subset is selected for training, and this is stepwise incremented with 20 examples. Training and testing for the points in the learning curve is repeated 10 times, and the mean is plotted.

Taguchi where ANN clearly outperforms the other algorithms. This shows that for t_R prediction boosting a decision tree forest (GB) performs significantly better than bagging a forest (RF). For completeness, the same observations for each algorithm are made with the mean absolute error and the correlation between observed and experimental retention times (Figure S2, S3 and S4).

3.2 Prediction performance on all datasets

In this section we use all 36 datasets to further investigate the performance differences between the different regression models. For each dataset we use

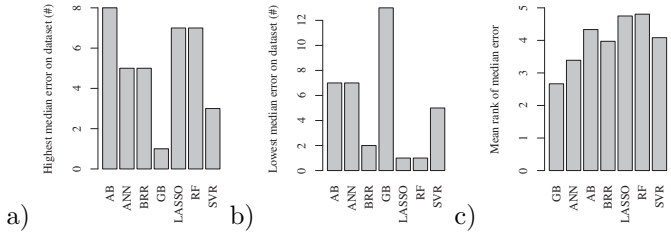

Figure 2: The frequency an algorithm scored the lowest (a) and highest (b) absolute median error on the folds of the CV per dataset. Panel (c) shows the mean rank over all datasets when the performance is sorted on absolute median error.

all the unique molecules to compile a feature vector dataset and apply 10-fold CV to assess the generalization performance. The CV results with median and mean absolute error can be found in Table S3 and S4.

Figure 2(a) shows the number of times an algorithm had the lowest median absolute error on any of the 36 datasets. GB again clearly sticks out here, being the best performing algorithm for 13 datasets. Linear models such as BRR and LASSO (but again also the non-linear RF) are the worst performing algorithms having the best performance on only one or two datasets.

Figure 2(b) shows the number of times an algorithm had the highest median absolute error on a dataset. Here we again observe how GB performs significantly better than the other algorithms, having the worst performance for just one dataset (UniToyama_Atlantis). The SVR (linear/gaussian) model follows closely, having the worst median error on three datasets. Notice how adaptive boosting is ranked worst for 8 out of the 36 datasets, while achieving best performance for 5 datasets. This comparison shows that GB will not always perform the best, but out of all algorithms it is most likely to show the best generalization performance and is least likely to show the worst performance.

Even though there is a clear difference between algorithms for the best and worst performance, the mean rank for algorithms is more similar (Figure 2(c)). The similarity in mean rank indicates that high performing algorithms (GB,

MLP, SVR) can perform mediocre but not the worst on some of the datasets. These results clearly show the importance of evaluating multiple algorithms on an individual basis per dataset.

3.3 Pairwise performance comparison

To get more insight into the robust performance of GB for t_R prediction we compared its performance against the other learning algorithms for each of the 36 datasets. Figure 3 shows the difference in median absolute errors normalized to the maximum elution times for each dataset (eqn. 4). GB improves the normalized median error by 0.74 % to 1.51 % on average for all datasets compared to the other six algorithms. For example, in the case of Taguchi a 1 % improvement would translate to 40.89 second lower median absolute error. The datasets Matsuura, Matsuura_15 and Beck are consistently a worst fit for GB when compared to the other algorithms. For completeness, the pairwise comparisons are also made for AB, BRR, ANN, Random Forest and SVR models (Figure S.4). The differences in normalized median absolute error ranges from 0.01 % to 0.77 % and are overall much lower than the differences observed in Figure 3 for GB.

Figure 3 shows a number of datasets where the median error is worse for GB (shown as negative values). Figure 4 shows the comparison of choosing the best performing algorithm compared to choosing GB for each dataset. For the thirteen datasets with zero increase in performance the GB model is chosen, but for the remaining datasets choosing the highest performing model improved the relative median error with 0.72 % on average (Figure 4(a)). The performance increase is even more obvious for the SVR with an average improved relative median error of 1.84 % (Figure 4(b)). This shows that evaluating multiple algorithms and selecting the best model can significantly lower the prediction error.

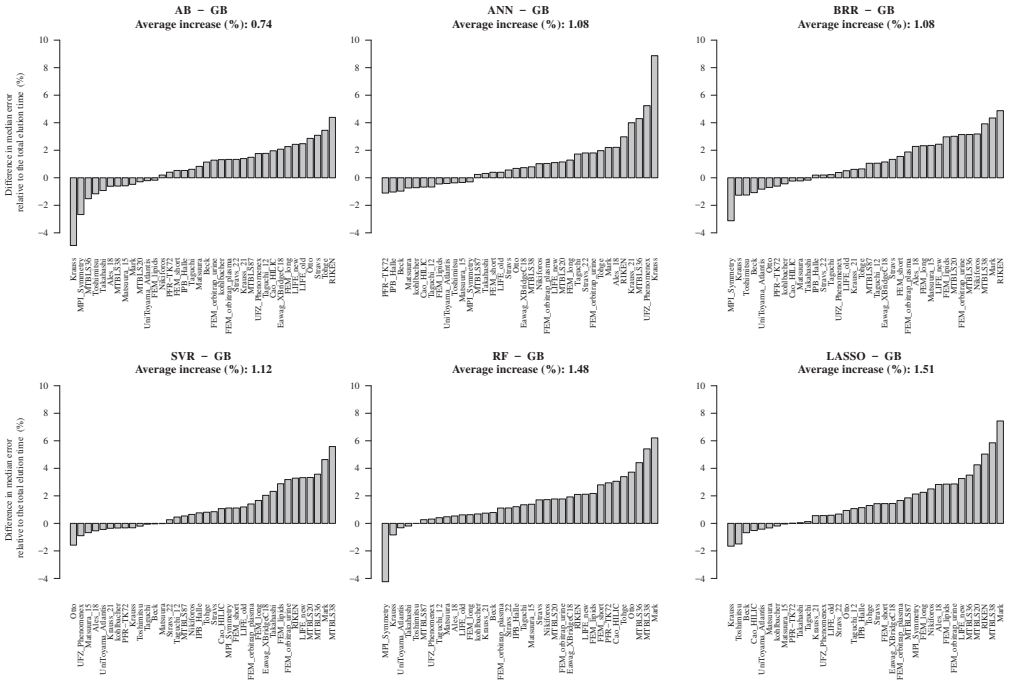

Figure 3: Difference in the CV absolute median error per dataset for the GB models compared with LASSO, AB, BRR, ANN and SVR. Positive numbers indicate that the median absolute error was lowered (improved) by the indicated amount when using GB. Negative numbers indicate a lower median absolute error for the algorithm GB is compared with.

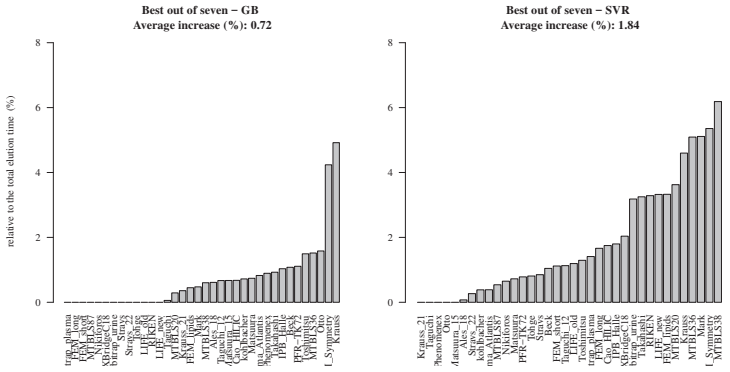

Figure 4: CV result comparison of the normalized median error of the GB and SVR models and best performing models out of the selected seven per dataset. Zero difference in the error indicates the algorithm under evaluation performed the best. Non-zero values indicate the lower error when choosing one of the other algorithms.

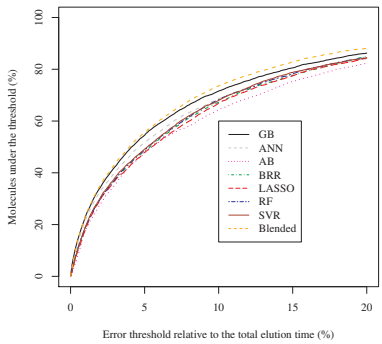

Figure 5: For an increasingly more lenient error threshold the percentage of metabolites across all 36 datasets under the threshold is calculated. Blended predictions are added by taking the average prediction for a metabolite of the models from GB, ANN and SVR.

3.4 Reducing prediction error outliers

Instead of selecting the best learning algorithm for a dataset one might also consider combining the predictions of several models to compute what is called blended predictions. This approach can only achieve better performance than the best performing model if the predictions computed by the different models are not too correlated. Figure S7 shows the correlation between the errors for the different learning algorithms. It shows that this correlation is high ($R > 0.82$) for algorithms that fit a continuous mathematical function (SVR, BRR and LASSO) and are very distinct from the tree based algorithms (GB, AB and RF) ($R < 0.59$) that between themselves also show a relatively high correlation ($R > 0.73$). Finally, the ANN model shows very low correlation with any of the other algorithms ($R < 0.58$).

Given these correlations we implemented a very simple blending strategy that averages the predictions of SVR, ANN and GB. The effect of blending will be minor for molecules that show similar t_R predictions for the three blended models. However for molecules that show sufficiently different tR predictions the potential effect of blending is that outlying predictions (i.e. predictions with large error) will be reduced.

To investigate this we looked at the percentage of molecules (over all 36 datasets) with prediction error lower than a certain threshold. Figure 5 shows different values for the threshold on the x-axis with the corresponding percentage of molecules with prediction error below the threshold on the y-axis. Figure 5 shows that the majority of metabolites are predicted with high accuracy. For GB the threshold where 50 % of the molecules shows lower absolute error is 4.1 %. After this point we can see that the error threshold increases rapidly. For example, for GB the threshold has to be set to 17.7 % to obtain 80 % of the metabolites with lower absolute error. Notice how the averaged (blended) predictions indeed perform better than the single best model for molecules with larger prediction error. For individual datasets both the mean and median absolute error rank is improved when blending and is better than the GB models

(Table S5). So, even though the difference between the blended model and GB is small, our results show that outlying t_R predictions can be reduced by a simple blending strategy. It is likely that more advanced blending strategies might reduce prediction error further.

3.5 Comparison SVR and gradient boosting models

The SVR algorithm is one of the most popular algorithms for t_R prediction, but our analysis shows that the GB algorithm is much more accurate for most datasets. The boosting algorithm in GB works by decreasing the bias of an ensemble of weak learners that have a low variance and high bias.The downside of this is that there should be enough training examples to effectively decrease the bias.

Figure 5 shows a comparison between the performance of both algorithms taking the size of the dataset into account. The 36 datasets are split into two equal sized groups with high and low number of training examples using 100 training examples as a seperator (the dotted horizontal line). For 13 out of 18 datasets in the group with a high number of training examples the median absolute error is lower for GB compared to the SVR. While for less than 100 training examples 9 out of 18 datasets have a lower median absolute error for GB; which is closer to randomly sampling datasets. The mean absolute error shows the same trend where under 100 traning examples 10 out of 18 datasets have a better performance for the SVR. Above 100 training examples only 3 out of 18 datasets have a better performance on the SVR dataset.

3.6 Reduced feature set

Overfitting is a problem for any machine learning task, but due to the relatively high amount of features (151) compared to the number of training examples this becomes a serious concern for some datasets. To detect overtraining on 151 features it is compared to models that were trained on a subset of eleven features where overtraining is less likely to occur.

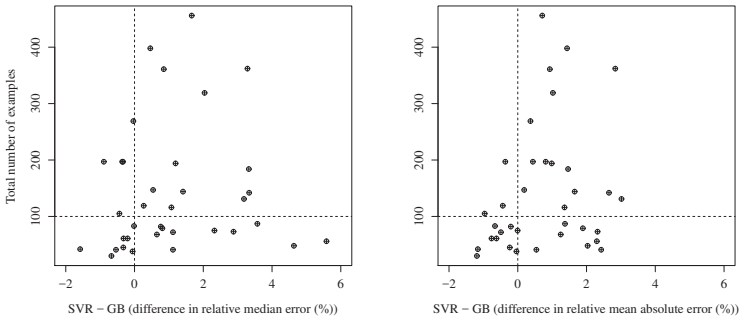

Figure 6: Comparison of the median and mean absolute error between GB and the SVR algorithm for every dataset represented by a datapoint. The difference in the median and mean absolute error between GB and SVR is plotted with the number of training examples. The vertical line indicates the area where there is no performance difference. The horizontal lines indicates a cut-off where GB frequently outperforms SVR models.

Figure 7 shows that the three datasets (Matsuura, Matsuura_15 and Beck) that were consistently a worse fit for GB in Figure 3 achieve a higher performance on the smaller pre-selected set of features. The higher performance on the smaller set of features indicates overtraining. However, limiting the number of features decreased the performance for most datasets (23 out of 36) with on average a lower median absolute error of 0.42 % for GB models. This shows that the bigger set of features is still preferred over the pre-selected smaller set of features for most datasets.

The pairwise comparison between the other learning algorithms that are fitted on the reduced feature set in Figure S7 follows the same conclusions as Figure 3. GB is able to outperform the other algorithms with an improved median absolute error of 0.94 % to 2.75 % on average. This shows that the earlier conclusions made with the complete feature set are not due to overtraining of some of the algorithms on the large set of features.

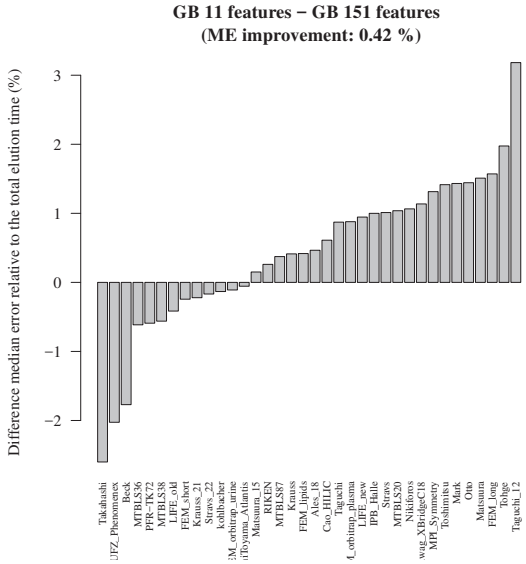

Figure 7: Difference in the absolute median error per dataset for GB with eleven or 151 features. The feature set was limited to eleven features that were used in [1]. Positive numbers mean that the absolute median error was lowered by the indicated amount when applying GB with 151 features.

3.7 Feature relevance

Our performance evaluations have shown that overall GB models perform best. In this section we investigate the relevance of each feature in all GB models. The F-score in GB reflects the feature relevance and is computed as the number of times a feature was selected to split the training examples during decision tree construction.

Figure 8 shows the highest scoring feature based on the F-score. The MolLogP describes the hydrophobicity and is the most important feature across all datasets. This is the result of the large proportion of datasets based on reverse-phase LC that utilizes hydrophobicity to separate the analytes. A large proportion of the important features describe the structural features of molecules (Chi4v, kappa, EState, PEOE and SlogP). The relevance of these features can be explained by interactions of reactive groups (e.g. cyclic carbons) with the column and similarity between molecules. High structural and chemical similarity between molecules seems to be a good indicator for molecules to have the same retention time.

Overall Figure 8 shows that importance of features is shared between datasets, but a significant proportion of features used in the models are shared by only a few datasets. These differences in feature importance underline that knowledge of features cannot be directly transferred between different experimental setups.

4 Conclusion

When making a LC t_R predictor researchers make multiple decisions that can influence the performance and interpretability of the final model. Most notable the number of training examples, the machine learning algorithm and molecular features.

The number of training examples should at least be 40 to get a model with fairly accurate predictions. However, different datasets and algorithms require a different number of training examples to achieve the highest performance

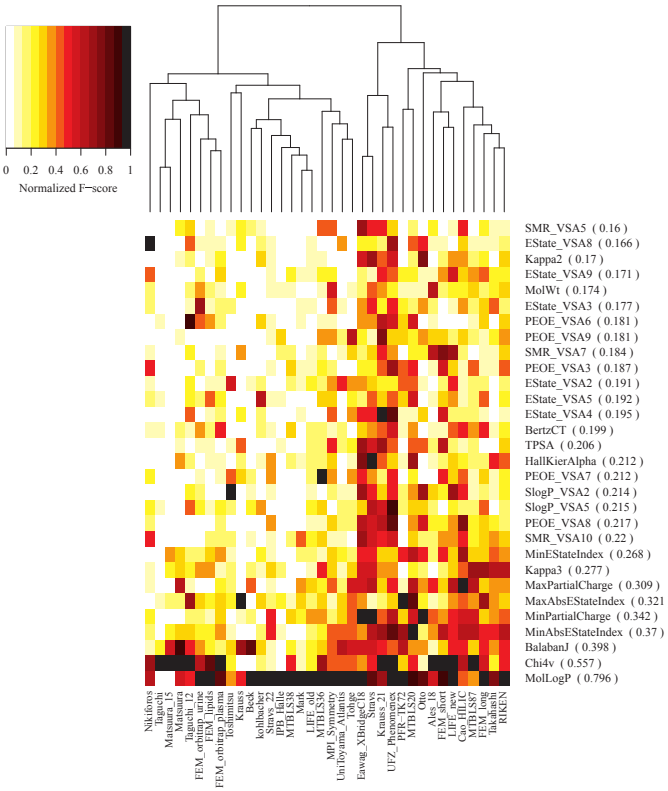

Figure 8: The normalized F-score from the GB algorithm per dataset. F-scores were normalized by dividing all the features by the maximum F-score per dataset. Every feature name is followed by the mean F-score of all datasets. Any feature with a mean F-score below 0.15 is excluded.

possible. Generally the GB algorithm gives the best performance with sufficient training examples. GB models generally keep improving for increasing number of training instances where other algorithms converge in their performance.

Cross-validation confirms that GB is the most likely candidate to give you the best performance and least likely to give the worst performance. However, all the other algorithms combined had the best performance in 23 out of 36 datasets. This shows the importance of testing different algorithms before choosing the learning algorithm that is used to generate the final model. GB, ANN and SVR generate complementary models and are a good initial start for evaluating multiple algorithms.

The feature importance of the GB models show that feature relevance is dataset dependent. Selecting features based on other experimental setups will result in a model that can not reach its full potential. In the initial stages of creating a model a researcher should be careful with excluding features. GB is generally able to select the features that are important to achieve a high performance without overtraining.

In conclusion, the difference in experimental setups means that there is not one best method to create a predictive model. Overall GB models outperform the models based on other algorithms and thus should at least be one of the candidate algorithms. Currently researchers make assumptions when creating a model; instead researcher should make decisions for an algorithm and features based on empirical evidence from independent performance evaluations.

Funding

This project was made possible by MASSTRPLAN. MASSTRPLAN received funding from the Marie Sklodowska-Curie EU Framework for Research and Innovation Horizon 2020, under Grant Agreement No. 675132.

References

[1] F Aicheler, J Li, M Hoene, R Lehmann, G Xu, and O Kohlbacher. Retention Time Prediction Improves Identification in Nontargeted Lipidomics Approaches. *Analytical Chemistry*, 87(15):7698–7704, aug 2015.

[2] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[3] Ian M Bird. High performance liquid chromatography: principles and clinical applications. *BMJ: British Medical Journal*, 299(6702):783, 1989.

[4] Paul G. Boswell, Jonathan R. Schellenberg, Peter W. Carr, Jerry D. Cohen, and Adrian D. Hegeman. Easy and accurate high-performance liquid chromatography retention prediction with different gradients, flow rates, and instruments by back-calculation of gradient and flow rate profiles. *Journal of Chromatography A*, 1218(38):6742–6749, 2011.

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Mingshu Cao, Karl Fraser, Jan Huege, Tom Featonby, Susanne Rasmussen, and Chris Jones. Predicting retention time in hydrophilic interaction liquid chromatography mass spectrometry and its use for peak annotation in metabolomics. *Metabolomics*, 11(3):696–706, 2015.

[7] Jie Chen, Ting Yang, and Steven M. Cramer. Prediction of protein retention times in gradient hydrophobic interaction chromatographic systems. *Journal of Chromatography A*, 1177(2):207–214, 2008.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794. ACM, 2016.

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[10] Darren J Creek, Andris Jankevics, Rainer Breitling, David G Watson, Michael P Barrett, and Karl E V Burgess. Toward global metabolomics analysis with hydrophilic interaction liquid chromatography–mass spectrometry: improved metabolite identification by retention time prediction. *Analytical chemistry*, 83(22):8703–8710, 2011.

[11] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[12] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[13] Hassan Golmohammadi, Zahra Dashtbozorgi, and Yvan Vander Heyden. Support Vector Regression Based QSPR for the Prediction of Retention Time of Peptides in Reversed-Phase Liquid Chromatography. *Chromatographia*, 78(1-2):7–19, jan 2015.

[14] A. E. Hoerl and R. W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, feb 1970.

[15] Aaron A Klammer, Xianhua Yi, Michael J MacCoss, and William Stafford Noble. Improving tandem mass spectrum identification using peptide retention time prediction across diverse chromatography conditions. *Analytical Chemistry*, 79(16):6111–6118, 2007.

[16] M Kuhn, J Wing, S Weston, and A Williams. caret: classification and regression training. R package version 6.0-24. 2014.

[17] G Landrum. The RDKit Documentation The RDKit 2016.09.1 documentation, 2016.

[18] Beilei Lei, Shuyan Li, Lili Xi, Jiazhong Li, Huanxiang Liu, and Xiaojun Yao. Novel approaches for retention time prediction of oligonucleotides in ion-pair reversed-phase high-performance liquid chromatography. *Journal of Chromatography A*, 1216(20):4434–4439, 2009.

[19] Wenyuan Lu, Xiaohui Liu, Shanshan Liu, Weiqian Cao, Yang Zhang, and Pengyuan Yang. Locus-specific Retention Predictor (LsRP): A Peptide Retention Time Predictor Developed for Precision Proteomics. *Scientific reports*, 7:43959, mar 2017.

[20] Feng Luan, Chunxia Xue, Ruisheng Zhang, Chunyan Zhao, Mancang Liu, Zhide Hu, and Botao Fan. Prediction of retention time of a variety of volatile organic compounds based on the heuristic method and support vector machine. *Analytica Chimica Acta*, 537(1):101–110, 2005.

[21] Chunwei Ma, Zhiyong Zhu, Jun Ye, Jiarui Yang, Jianguo Pei, Shaohang Xu, Ruo Zhou, Chang Yu, Fan Mo, Bo Wen, and Others. DeepRT: deep learning for peptide retention time prediction in proteomics. *arXiv preprint arXiv:1705.05368*, 2017.

[22] Rishabha Malviya, Vipin Bansal, Om Prakash Pal, and Pramod Kumar Sharma. High performance liquid chromatography: A short review. *Journal of Global Pharma Technology*, 2:22–26, 2010.

[23] L Mason, J Baxter, PL Bartlett, and MR Frean. Boosting Algorithms as Gradient Descent. *NIPS*, 1999.

[24] Wes McKinney. pandas: a Foundational Python Library for Data Analysis and Statistics. *Python for High Performance and Scientific Computing*, pp. 1–9, 2011.

[25] L Moruz and L Käll. Peptide retention time prediction. *Mass spectrometry reviews*, 2016.

[26] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and Others. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[27] Minghu Song, Curt M Breneman, Jinbo Bi, Nagamani Sukumar, Kristin P Bennett, Steven Cramer, and Nihal Tugcu. Prediction of protein retention times in anion-exchange chromatography systems using support vector regression. *Journal of chemical information and computer sciences*, 42(6):1347–1357, 2002.

[28] Vic Spicer, Andriy Yamchuk, John Cortens, Sandra Sousa, Werner Ens, Kenneth G Standing, John A Wilkins, and Oleg V Krokhin. Sequence-specific retention calculator. a family of peptide retention time prediction algorithms in reversed-phase hplc: applicability to various chromatographic conditions and columns. *Analytical chemistry*, 79(22):8762–8768, 2007.

[29] Jan Stanstrup, Steffen Neumann, and Urška Vrhovšek. PredRet: Prediction of Retention Time by Direct Mapping between Multiple Chromatographic Systems. *Analytical Chemistry*, 87(18):9421–9428, sep 2015.

[30] Eric F Strittmatter, Lars J Kangas, Konstantinos Petritis, Heather M Mottaz, Gordon A Anderson, Yufeng Shen, Jon M Jacobs, David G Camp, and Richard D Smith. Application of peptide LC retention time information in a discriminant function for peptide identification by tandem mass spectrometry. *Journal of proteome research*, 3(4):760–769, 2004.

[31] R Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (*, 1996.

[32] David Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[33] Ian D Wilson, Robert Plumb, Jennifer Granger, Hilary Major, Rebecca Williams, and Eva M Lenz. Hplc-ms-based methods for the study of metabonomics. *Journal of Chromatography B*, 817(1):67–76, 2005.

[34] Arnaud M. Wolfer, Sylvain Lozano, Thierry Umbdenstock, Vincent Croixmarie, Alban Arrault, and Philippe Vayer. UPLCMS retention time prediction: a machine learning approach to metabolite identification in untargeted profiling. *Metabolomics*, 12(1):8, jan 2016.

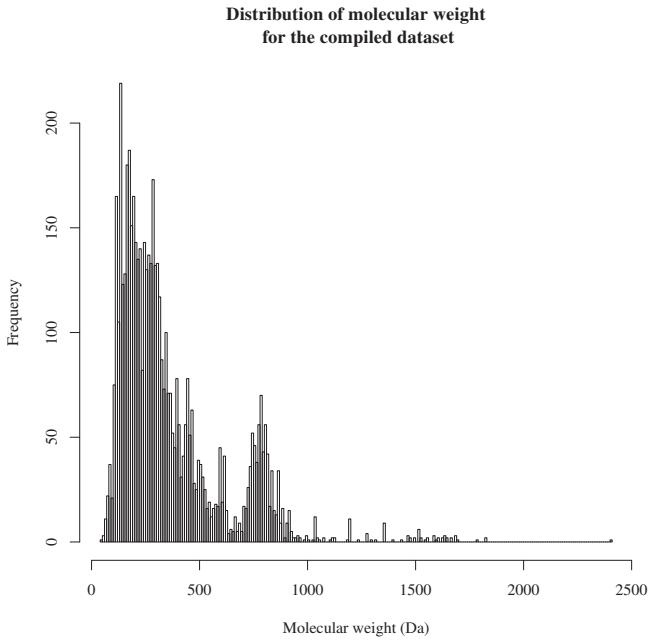

Figure S1: Molecular weight distribution for metabolites used across all 36 datasets.

Table S1: Properties of the different datasets used.

Dataset	Molecules	Highest MW	Lowest MW	Highest MolLogP	Lowest MolLogP
Krauss	61	288.782	69.067	6.2066	-0.1953
Nikiforos	68	916.112	114.173	6.80734	-1.0397
Vogler	5	314.469	102.097	4.7235	-1.03776
Kojima	37	513.504	175.187	3.8441	-3.3875
Tohge	79	740.664	224.212	3.3931	-2.892
Otto	42	563.672	136.114	6.1213	-1.98
FEM_short	72	906.94	74.079	10.4322	-3.0658
LIFE_new	184	834.173	75.111	12.3886	-5.3972
LIFE_old	194	826.109	44.078	11.4926	-5.3972
IPB_Halle	82	521.676	112.088	5.1358	-2.44403
MTBLS39	34	638.578	134.087	1.5461	-5.3956
MTBLS87	147	911.667	105.093	4.4779	-5.3972
MTBLS4	34	836.189	467.584	12.6126	3.7994
FEM_lipids	73	891.501	144.214	18.7691	-1.8065
FEM_orbitrap_plasma	144	786.129	77.152	11.7242	-2.4396
Cao_HILIC	116	663.43	59.068	3.8257	-5.6689
FEM_orbitrap_urine	131	515.713	60.052	7.8929	-3.5854
MTBLS17	26	567.704	161.126	5.5762	1.851
MTBLS19	28	584.673	131.135	8.9599	-1.23283
MTBLS52	29	860.771	386.353	3.07084	-3.8778
MPI_Symmetry	41	822.953	137.182	5.134	-2.86343
1290SQ	8	362.466	133.194	4.3806	-0.6459
PFR-TK72	45	1034.2	154.121	7.2336	-2.6973
MTBLS38	56	351.363	111.104	5.3033	-5.3972
MTBLS36	87	504.438	88.062	4.0662	-7.5698
Mark	48	351.363	111.104	5.3033	-3.2602
Toshimitsu	61	740.664	256.257	7.2336	-2.892
Matsuura	83	1828.691	364.351	1.9792	-20.484
UniToyama_Atlantis	105	1109.307	222.196	7.2336	-2.892
Ales_18	41	822.953	137.182	5.134	-2.86343
Ken	25	957.117	222.196	4.6299	-1.5149
Matsuura_15	38	1518.337	545.491	-6.753	-20.484
Stravs	361	995.189	119.127	11.919	-1.6561
Beck	269	1045.205	74.083	7.8493	-3.0115
RIKEN	362	932.834	73.095	7.5065	-7.573
Takahashi	75	809.578	103.121	1.1223	-3.8768
Krauss_21	197	837.058	113.12	6.718	-1.03416
UFZ_Phenomenex	197	837.058	113.12	6.718	-1.03416
FEM_long	456	1272.453	59.072	11.168	-9.7472
Taguchi	38	907.459	663.037	18.3211	12.0795
kohlbacher	197	915.196	172.268	13.6503	2.3565
MTBLS20	142	521.676	60.056	5.1358	-5.1426
Huntscha	22	503.515	69.067	5.458	-1.2349
Eawag_XBridgeC18	319	995.189	119.127	7.1543	-1.6561
Stravs_22	119	785.891	129.159	7.1837	-1.0397
Taguchi_12	398	2406.648	437.558	39.6054	1.6921

Table S2: List of all molecular descriptors (features) calculated using rdkit to predict the retention time.

fr_C_O_noCOO	fr_Al_COO	SlogP_VSA5	HeavyAtomCount
PEOE_VSA3	EState_VSA10	SlogP_VSA6	fr_azide
Chi4v	EState_VSA11	SlogP_VSA7	NumHAcceptors
fr_Ar_COO	HeavyAtomMolWt	SlogP_VSA1	fr_lactam
fr_SH	fr_nitro_arom	SlogP_VSA2	fr_allylic_oxid
Chi4n	Chi0	SlogP_VSA3	VSA_EState8
SMR_VSA10	Chi1	NumRadicalElectrons	fr_oxazole
fr_para_hydroxylation	NumAliphaticRings	fr_NH2	VSA_EState9
fr_barbitur	MolLogP	fr_piperzine	fr_piperdine
fr_Ar_NH	fr_nitro	fr_nitrile	fr_Ar_OH
fr_halogen	fr_Al_OH	NumHeteroatoms	fr_sulfide
fr_dihydropyridine	fr_azo	fr_NH1	fr_alkyl_carbamate
fr_priamide	NumAliphaticCarbocycles	fr_NH0	NOCount
SlogP_VSA4	fr_C_O	MaxAbsEStateIndex	Chi1n
fr_guanido	fr_ether	LabuteASA	PEOE_VSA8
MinPartialCharge	fr_phenol_noOrthoHbond	fr_amide	PEOE_VSA7
fr_furan	fr_alkyl_halide	Chi3n	PEOE_VSA6
fr_morpholine	NumValenceElectrons	fr_imidazole	PEOE_VSA5
fr_nitroso	fr_aryl_methyl	SMR_VSA3	PEOE_VSA4
NumAromaticCarbocycles	fr_Ndealkylation2	SMR_VSA2	MaxEStateIndex
fr_COO2	MinEStateIndex	SMR_VSA1	PEOE_VSA2
fr_amidine	fr_term_acetylene	Chi3v	PEOE_VSA1
SMR_VSA7	HallKierAlpha	SMR_VSA6	NumSaturatedCarbocycle
fr_benzodiazepine	fr_C_S	Kappa3	fr_imide
ExactMolWt	fr_thiocyan	Kappa2	FractionCSP3
fr_Imine	fr_ketone_Topliss	EState_VSA6	Chi1v
MolWt	VSA_EState4	EState_VSA7	fr_Al_OH_noTert
fr_hdrzine	Ipc	SMR_VSA9	fr_epoxide
fr_urea	VSA_EState6	EState_VSA5	fr_hdrzone
NumAromaticRings	VSA_EState7	EState_VSA2	fr_isothiocyan
fr_quatN	VSA_EState1	EState_VSA3	NumAromaticHeterocycle
NumSaturatedHeterocycles	VSA_EState2	fr_Ndealkylation1	fr_bicyclic
NumAliphaticHeterocycles	VSA_EState3	EState_VSA1	Kappa1
fr_benzene	fr_HOCCN	fr_ketone	Chi0n
fr_phos_acid	fr_phos_ester	SMR_VSA5	fr_phenol
fr_sulfone	BertzCT	MinAbsEStateIndex	MolMR
VSA_EState10	SlogP_VSA12	fr_diazo	PEOE_VSA9
fr_aniline	EState_VSA9	SMR_VSA4	fr_aldehyde
fr_N_O	SlogP_VSA10	fr_Ar_N	fr_pyridine
fr_sulfonamd	SlogP_VSA11	fr_Nhpyrrole	fr_tetrazole
fr_thiazole	fr_COO	fr_ester	RingCount
TPSA	NHOHCount	VSA_EState5	fr_nitro_arom_nonortho
EState_VSA8	fr_unbrch_alkane	EState_VSA4	Chi0v
PEOE_VSA14	NumSaturatedRings	NumHDonors	fr_ArN
PEOE_VSA13	MaxPartialCharge	fr_prisulfonamd	NumRotatableBonds
PEOE_VSA12	fr_methoxy	fr_oxime	MaxAbsPartialCharge
PEOE_VSA11	fr_thiophene	SMR_VSA8	
PEOE_VSA10	SlogP_VSA8	fr_isocyan	
BalabanJ	SlogP_VSA9	Chi2n	
fr_lactone	MinAbsPartialCharge	Chi2v	

Table S3: The median absolute error in seconds split out for every algorithm and dataset.

Experiment	GB	ANN	AB	BRR	LASSO	RF	SVR
FEM_orbitrap_plasma	35.4	50.9	55.3	63.4	60.2	52.0	56.4
MTBLS38	62.4	68.3	57.9	91.6	106.0	102.7	103.9
FEM_long	117.3	164.5	200.4	202.7	200.1	140.5	178.4
FEM_lipids	57.4	50.7	54.8	101.8	100.0	89.9	100.4
UniToyama_Atlantis	75.8	67.8	71.7	59.9	67.7	69.6	67.3
FEM_short	91.8	97.0	98.8	112.2	110.7	128.7	106.5
MTBLS87	67.9	71.5	90.0	83.5	95.5	71.9	75.9
Takahashi	41.7	46.0	29.1	39.4	42.3	39.2	73.3
Krauss_21	115.3	191.1	141.8	126.9	126.0	129.4	108.6
Taguchi	19.9	90.3	45.0	29.2	25.2	75.2	17.5
MTBLS20	37.3	45.6	35.3	59.0	68.0	50.1	61.3
UFZ_Phenomenex	117.2	216.5	150.6	124.5	128.0	123.1	100.3
Taguchi_12	66.6	45.7	121.7	99.5	100.0	79.4	80.9
IPB_Halle	19.9	12.4	23.7	21.3	28.1	28.6	25.4
Nikiforos	39.7	48.4	41.4	66.8	61.1	54.4	45.3
Eawag_XBridgeC18	61.3	68.5	81.8	72.6	75.5	80.3	81.4
Ales_18	178.6	203.1	171.8	203.8	209.9	184.5	172.6
Beck	83.5	70.2	99.1	68.7	74.2	94.2	83.0
kohlbacher	9.2	4.9	17.1	6.6	8.1	13.3	7.2
FEM_orbitrap_urine	37.5	57.7	51.8	72.7	69.6	57.4	73.2
Stravs	57.7	64.6	95.4	74.0	75.2	78.6	68.1
Otto	101.7	109.6	135.0	93.5	112.6	145.0	83.3
Toshimitsu	74.4	67.4	52.3	50.6	46.1	74.4	70.6
Stravs_22	76.2	98.2	92.5	78.6	84.4	89.8	79.4
Matsuura_15	184.4	175.2	168.5	248.3	183.2	222.2	166.1
Tohge	90.1	117.6	138.3	99.1	108.2	137.5	101.4
Cao_HILIC	57.5	51.1	76.3	55.3	52.6	86.8	67.8
Matsuura	68.1	47.9	90.8	62.0	59.1	81.3	67.7
LIFE_old	9.4	10.8	17.8	11.1	11.4	11.5	13.4
MPI_Symmetry	154.8	151.5	125.2	120.2	178.4	107.8	167.2
Krauss	12.5	24.2	6.0	10.8	10.3	11.4	12.1
MTBLS36	68.8	103.6	56.6	94.2	97.2	104.5	97.7
RIKEN	4.5	8.4	10.2	10.8	11.0	7.3	8.8
Mark	56.3	72.6	52.8	88.6	111.6	102.5	90.8
LIFE_new	13.0	16.6	21.0	21.1	23.8	20.0	24.0
PFRTK72	41.8	29.0	46.5	34.8	41.9	75.9	38.0

Table S4: The mean absolute error in seconds split out for every algorithm and dataset.

Experiment	GB	ANN	AB	BRR	LASSO	RF	SVR
FEM_orbitrap_plasma	64.8	102.4	80.2	87.9	87.7	73.1	89.5
MTBLS38	95.0	106.4	97.2	103.1	122.6	112.9	112.1
FEM_long	248.4	254.7	333.6	283.6	1514.52	251.1	274.6
FEM_lipids	105.9	118.2	95.4	154.2	134.2	119.7	140.6
UniToyama_Atlantis	106.7	109.6	108.3	90.7	98.0	104.8	88.2
FEM_short	158.4	166.2	154.0	149.7	160.1	164.6	151.9
MTBLS87	114.0	108.7	124.7	115.2	124.3	117.6	116.8
Takahashi	92.1	80.6	82.2	65.2	67.9	86.4	92.0
Krauss_21	159.5	248.9	182.0	173.0	176.2	172.8	167.9
Taguchi	40.3	114.3	78.5	39.1	38.3	95.8	38.7
MTBLS20	56.3	62.0	59.9	72.4	74.0	68.4	75.4
UFZ_Phenomenex	156.8	259.5	183.2	173.9	178.0	164.7	172.2
Taguchi_12	97.2	72.2	160.2	154.0	1543.4	113.9	141.8
IPB_Halle	33.9	24.6	37.6	33.9	39.8	42.2	32.4
Nikiforos	64.0	95.3	67.4	84.8	77.6	72.3	74.7
Eawag_XBridgeC18	87.6	93.4	99.4	93.7	94.9	96.8	97.7
Ales_18	212.5	216.9	222.2	227.5	232.6	203.7	239.4
Beck	97.1	93.5	111.3	92.6	92.0	114.4	102.1
kohlbacher	12.7	6.2	19.7	9.9	10.9	17.6	10.6
FEM_orbitrap_urine	60.4	85.0	73.1	94.9	94.8	77.2	94.2
Stravs	81.4	89.2	107.7	94.1	96.4	99.0	92.8
Otto	144.6	177.7	155.1	136.5	153.6	173.8	131.0
Toshimitsu	104.8	141.9	97.5	93.4	84.2	102.5	93.1
Stravs_22	118.5	135.1	115.8	110.4	113.9	125.5	113.2
Matsuura_15	317.9	288.8	329.4	324.6	287.8	305.7	285.6
Tohge	118.7	155.8	157.7	127.5	130.4	169.9	145.1
Cao_HILIC	106.5	111.4	121.0	101.1	93.6	117.6	119.5
Matsuura	117.3	89.8	125.7	93.9	92.9	114.5	99.1
LIFE_old	14.8	17.0	19.6	17.0	17.4	16.0	18.1
MPI_Symmetry	181.2	216.1	188.6	175.1	222.3	178.9	187.2
Krauss	15.3	25.1	12.1	13.4	13.1	13.0	14.3
MTBLS36	117.9	162.5	115.7	127.3	133.9	130.7	129.0
RIKEN	9.8	13.7	13.9	14.5	14.8	11.6	13.5
Mark	93.0	93.1	94.2	104.3	134.3	109.0	108.1
LIFE_new	26.4	28.8	30.2	28.3	29.8	26.7	31.2
PFR-TK72	72.6	54.4	81.6	68.3	81.4	111.9	69.9

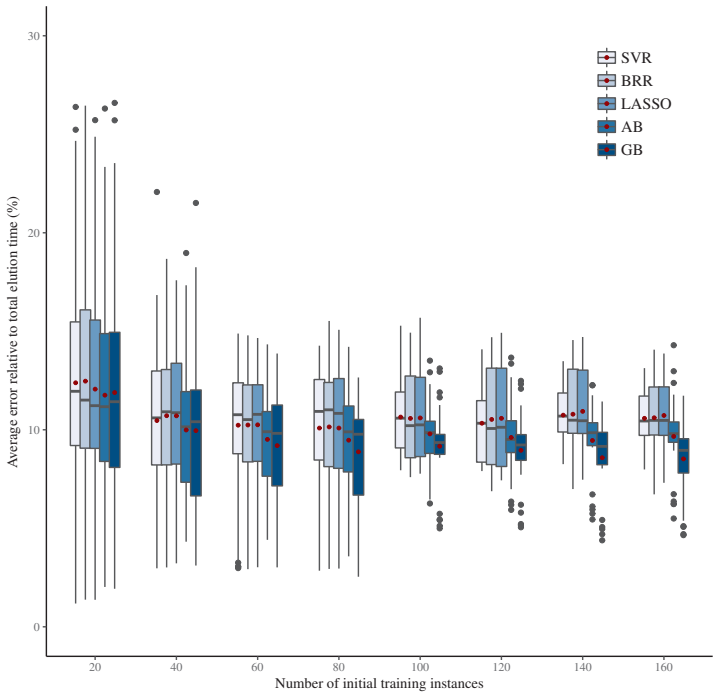

Figure S2: The absolute median error learning curves for training examples from 20 to 160 with steps of 20. The performance of five datasets for 10 repeats are aggregated in a single plot. The error in seconds varies per dataset and therefore they were normalized by the maximum elution time of a metabolite that was identified.

Table S5: Ranking of all machine learning algorithms, including blended predictions, in the 36 datasets. Blending was performed by taking the mean between GB, SVR and ANN predictions. As a metric both the mean and median absolute error are used.

Algorithm	Mean rank of mean absolute error	Mean rank of median absolute error
Blended	2.97	1.94
GB	3.11	3.47
BRR	4.69	4.42
SVR	4.86	4.89
ANN	4.08	5.06
LASSO	5.61	5.25
RF	5.67	5.28
AB	5.00	5.69

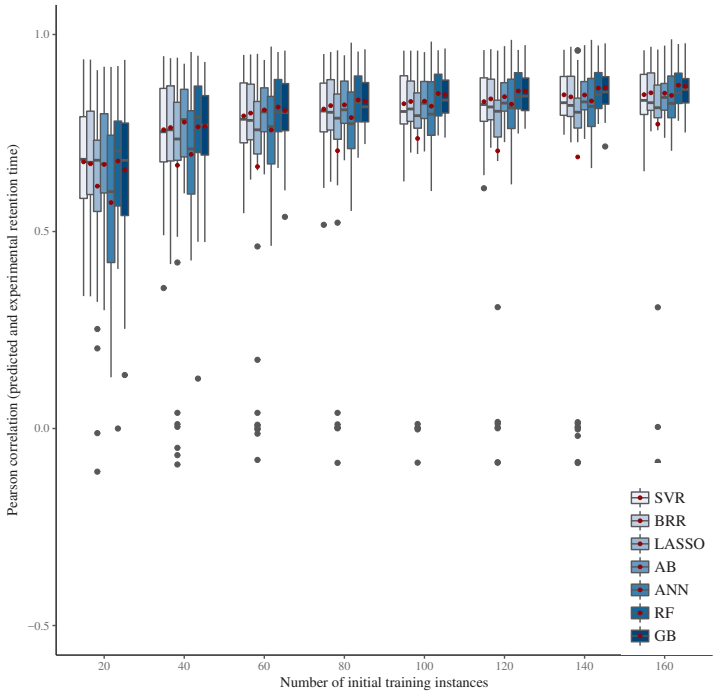

Figure S3: The pearson correlation learning curves for training examples from 20 to 160 with steps of 20. The performance of five datasets for 10 repeats are aggregated in a single plot. The error in seconds varies per dataset and therefore they were normalized by the maximum elution time of a metabolite that was identified.

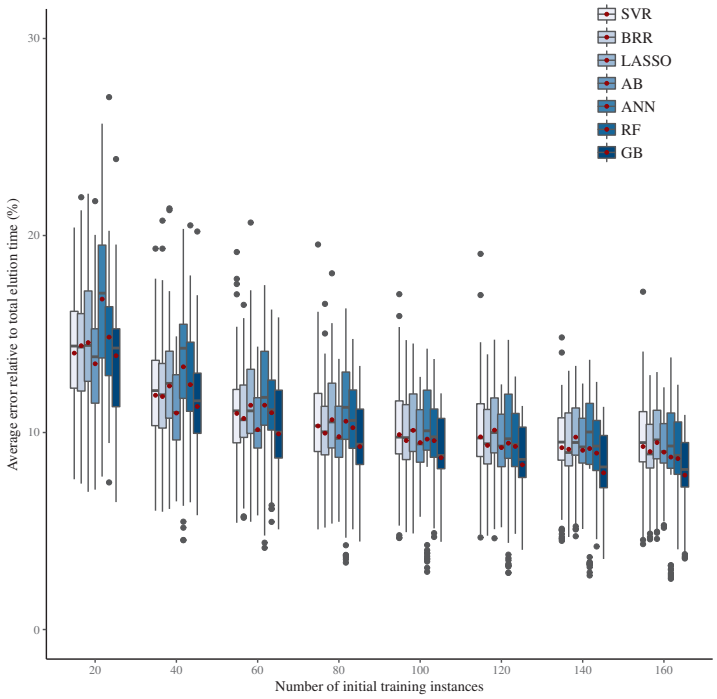

Figure S4: The mean absolute error learning curves for training examples from 20 to 160 with steps of 20. The performance of five datasets for 10 repeats are aggregated in a single plot. The error in seconds varies per dataset and therefore they were normalized by the maximum elution time of a metabolite that was identified.

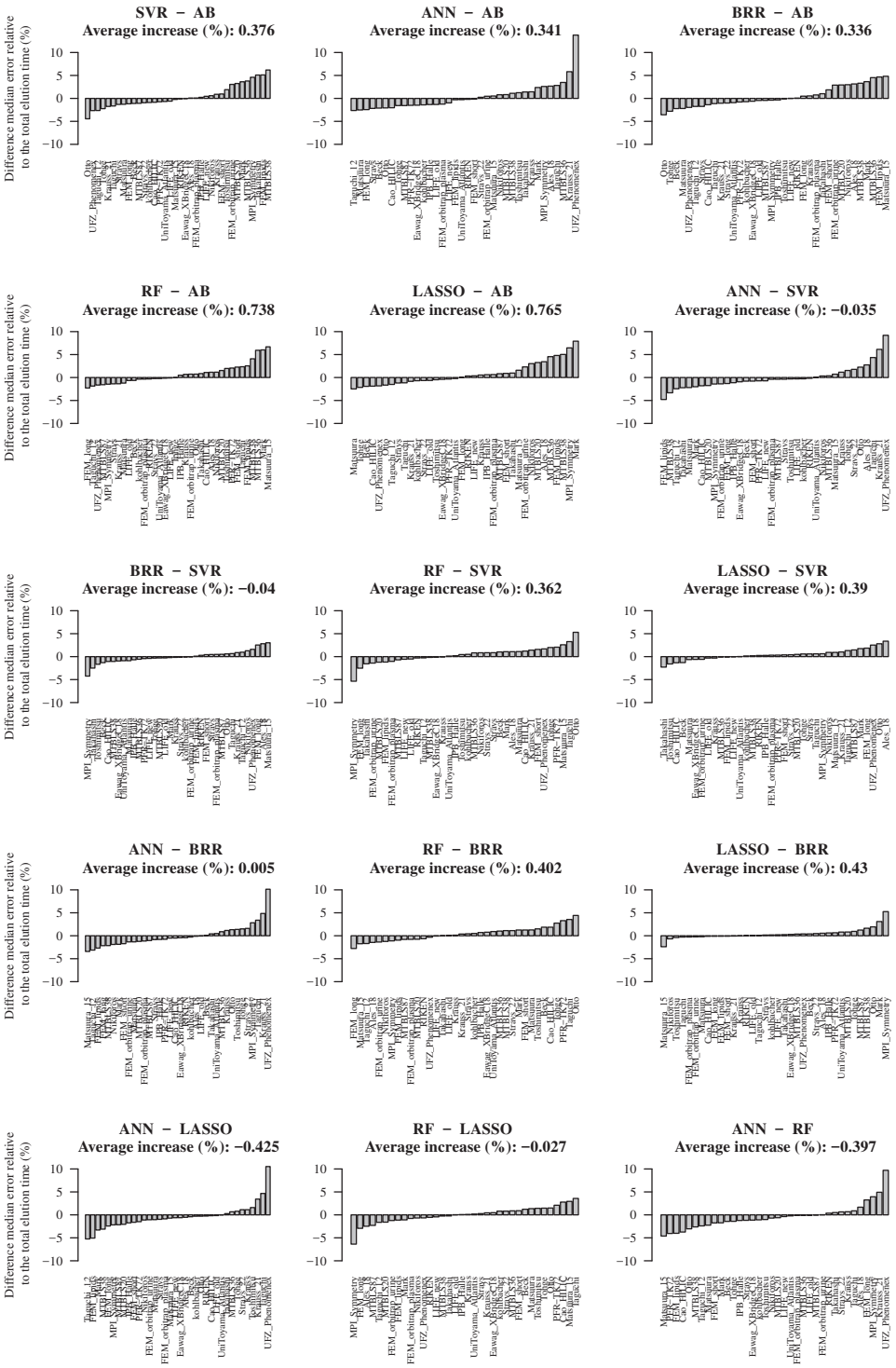

Figure S5: Difference in the absolute median error per dataset for four algorithms: Lasso, adaptive boosting, BRR and SVR. Positive numbers mean that the absolute median error was lowered by the indicated amount when using the algorithm that was first mentioned in the title.

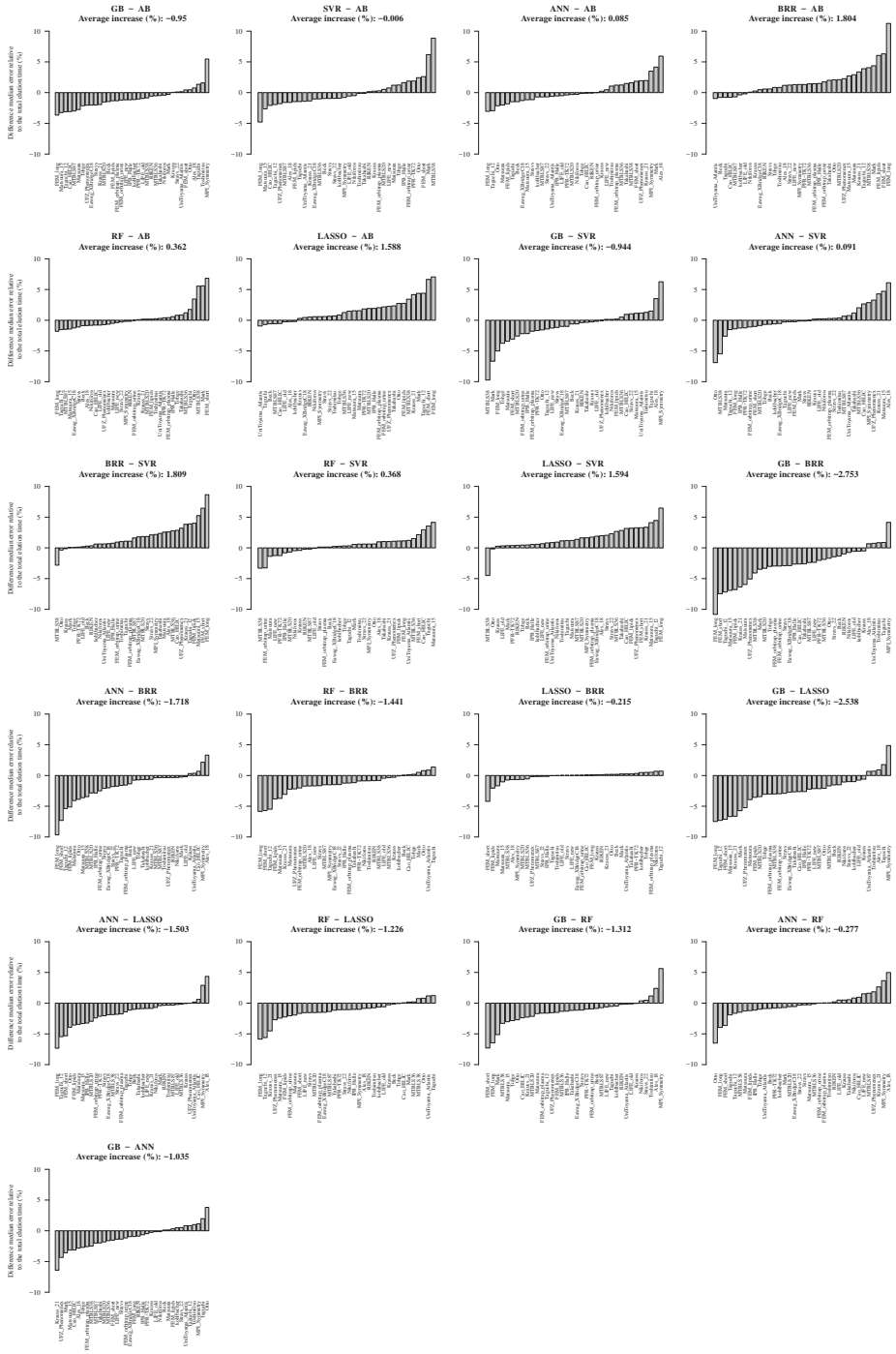

Figure S6: Difference in the absolute median error per dataset for five algorithms: gradient boosting, Lasso, adaptive boosting, BRR and SVR. In this comparison the feature set was limited to eleven features that were used in [1].Positive numbers mean that the absolute median error was lowered by the indicated amount when using the algorithm that was first mentioned in the title.

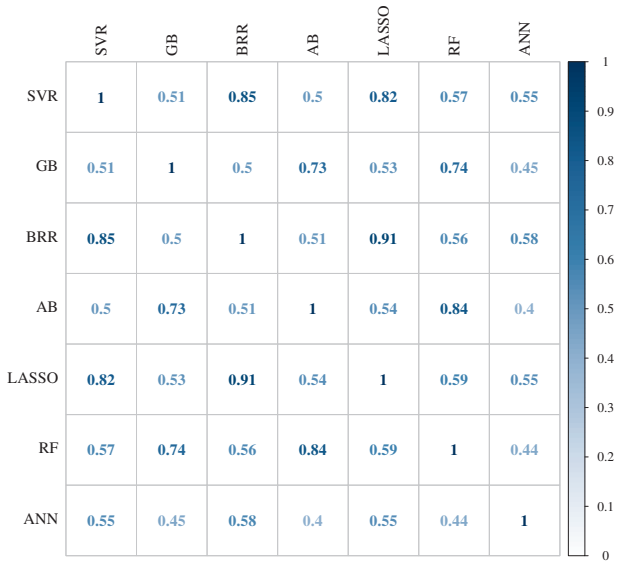

Figure S7: Correlation between the errors of all molecules in the 36 datasets and the seven machine learning algorithms. The number between the algorithm indicates the pearson correlation between the errors.

Listing 1: Distributions of hyperparameters used to optimize the regression models

```
# LASSO

params = {
'alpha' : uniform(0.0,10.0),
'copy_X' : [True],
'fit_intercept' : [True,False],
'normalize' : [False],
'precompute' : [True,False],
}

# Adaptive boosting
params = {
'n_estimators': randint(10,200),
'learning_rate': uniform(0.01,0.5)
}

# Gradient boosting

params = {
'n_estimators' : randint(10,200),
'max_depth' : randint(1,12),
'learning_rate' : uniform(0.01,0.25),
'gamma' : uniform(0.0,10.0),
'reg_alpha' : uniform(0.0,10.0),
'reg_lambda' : uniform(0.0,10.0)
}

# SVR

params = {
'C': uniform(0.01,300.0),
'epsilon': uniform(0.01,300.0),
'gamma' : [0.001,0.005,0.01,0.05,0.1,0.5,1.0,10.0,100.0],
'kernel': ['linear','rbf']
}

# BRR

params = {
'n_iter' : randint(200,800),
'alpha_1' : uniform(1e-7,1e-3),
'lambda_1' : uniform(1e-7,1e-3)
}

# Random Forest
```

```
params = {
'n_estimators': randint(1,250),
'max_features': randint(1,40),
'min_samples_leaf'  : randint(2,40),
'max_depth' : randint(1,10)
}

# ANN

params = {
'alpha' : logspace(-6.0,1.0),
'solver' : ['adam'],
'max_iter' : randint(25,100),
'hidden_layer_sizes' : randint(5,50)
}
```