Predicting the Chromatographic Retention Time of Small Molecules

Capstone Project Report

Tero Jalkanen

01/2022

Contents

1 Introduction

Liquid chromatography (LC) is an analytical laboratory method widely used with organic small molecules. LC can be used for separating different molecules in a mixture, or in an analytical way, where the aim is to detect the presence or relative proportion of molecules in a mixture.

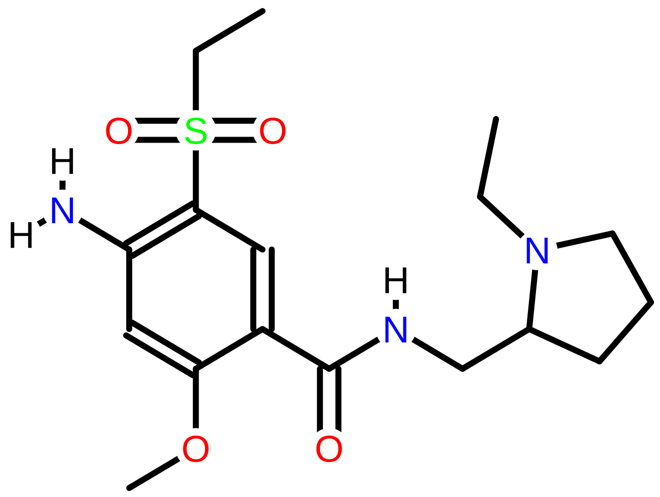

Figure 1: The chemical structure of one example molecule chosen from the original dataset.

In a LC measurement small molecules are dissolved in a fluid. The fluid with the small molecules (i.e. the analytes) is called the "mobile phase," and is washed through an LC column. The molecules in the mobile phase usually have an affinity towards the surface of the LC column, and are adsorbed or bound which slows them down, whereas the fluid carrying the molecules, the eluent, is washed through. Depending on the structure of the analyte molecule and the type of LC column, the molecules are retained in the column for differing amounts of time before exiting the column. This time is called the retention time, and it can be used, for example, in estimating the amounts of different molecules in a mixture.

The prediction of retention time is not a straightforward task. Among other thing the retention time depends on the type of LC column, and the structure of the molecule. Moreover, things such as the age of the column have an effect on the measured retention time. The METLIN small molecule dataset contains the retention times for over 80,000 molecules, measured with reverse-phase liquid chromatography (Domingo-Almenara et al. 2019). The dataset is freely available, and it was created for the purpose of enabling machine-learning based retention time prediction (Domingo-Almenara 2019). Domingo-Almenara *et al.* have demonstrated the utility of this dataset by predicting the retention time with a deep-learning model utilizing the calculated molecular fingerprints of the small molecules (Domingo-Almenara et al. 2019).

Here we will explore if using a set of simple molecular descriptors could be used for the same purpose.

1.1 Structure of the data

The original dataset contains three columns, namely the measured retention time in seconds, the molecular structure of the small molecule in the form of InChI textual identifier (*International Chemical Identifier*), and the *PubChem* identification number (Domingo-Almenara 2019). Figure 1 shows an example of one of the molecules in the dataset. These three columns alone do not give us much to work with. In the original paper, the authors used elaborate molecular fingerprints for retention time prediction. Here we aim for a differing approach, where as a preparatory task we have calculated several molecular descriptors for each molecule from the molecular structure. These desciptors, along with the retention time, will be used as our final dataset for this project.

1.1.1 Distribution of retention time values

The distribution of retention times is shown in Figure 2. The molecules which come through during the first 5 minutes or 300 seconds can be though as not having any specific affinity towards the LC column. They exit the column directly with the eluent. In this case, the retention time can not be used for laboratory analytical purposes.

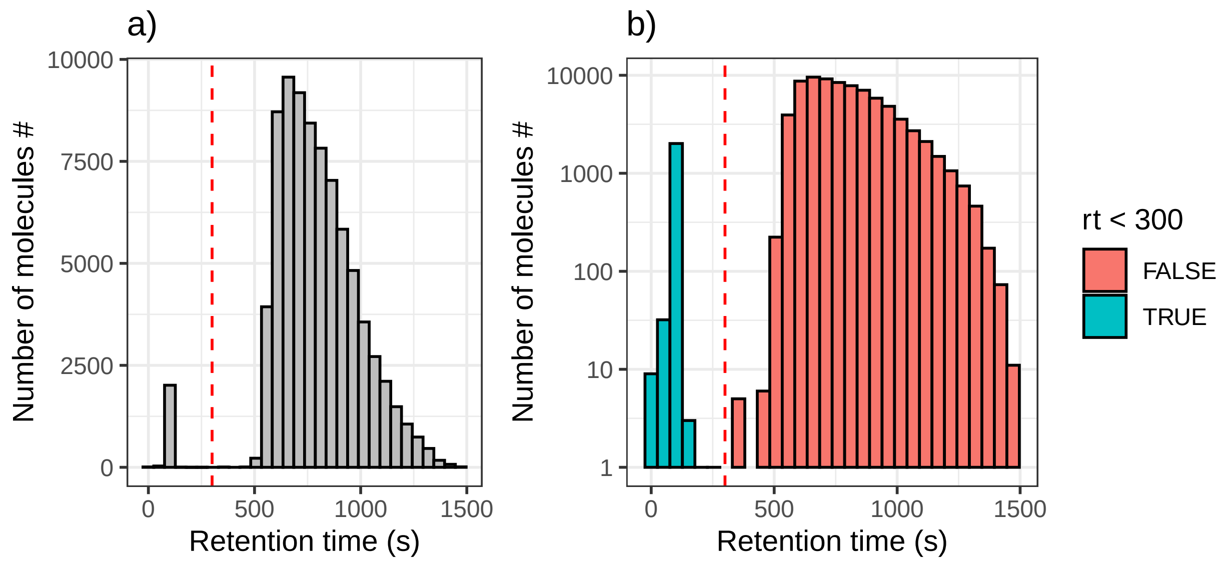

Figure 2: Histogram of retention times with a) linear and b) logarithmic y-axis. The molecules which are not retained in the LC column can be seen below 300 s. They are basically washed out with eluent, and are not very useful for analytical purposes.

1.2 Aims of the project

This project has two aims:

- First objective is to investigate if molecular descriptors such as physico-chemical properties can be used to accurately classify the molecules with regard to their retention class. That is whether a given molecule is retained or not retained in the LC column.

- The second objective is to try to predict the retention time for the retained molecules based on the molecular properties.

The first objective is interesting in the sense that it can help us identify molecules which are not suitable for a certain LC method. However, the main objective of this project is to try to estimate the retention time of retained small molecules.

2 Methods

2.1 Data preparation

As we discussed above, the original dataset contained only three columns (Domingo-Almenara 2019). However, as a preparatory step, several molecular descriptors have been calculated for the molecules using two cheminformatics `R` packages, namely the *ChemmineR* (Y. Cao et al. 2008) and *Rcpi* (D.-S. Cao et al. 2014) packages. The code for creating the dataset with the molecular descriptors is provided in the GitHub repository in a script file called `CreateDataset.R`. The final dataset contains 46 columns and 80038 observations.

2.2 Model training & evaluation

Model training for the classification and regression tasks will utilize a test/train split. Moreover, *k*-fold cross-validation will be utilized in the training phase for some models. The re-sampling methodology is further elaborated in Figure 3.

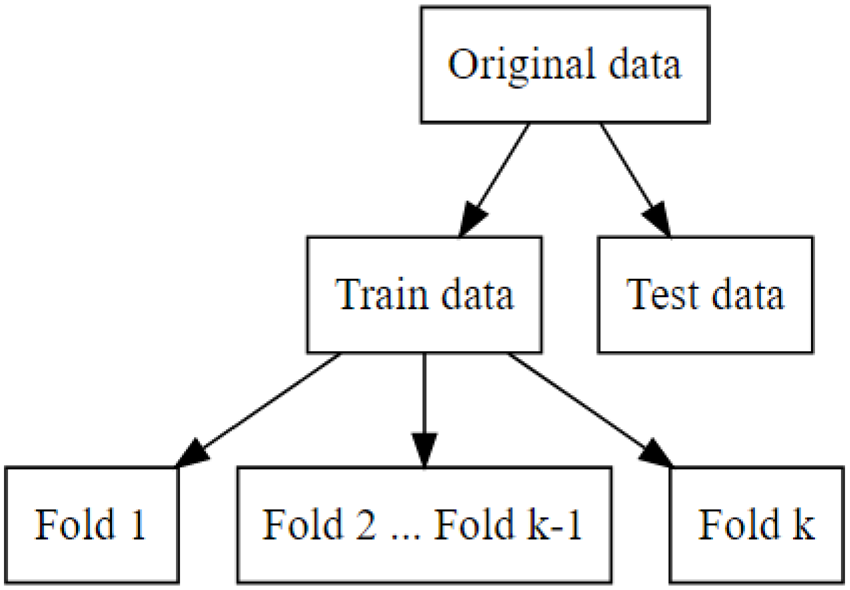

Figure 3: Splitting of data for training and testing purposes. Training data is further split into *k* roughly equally sized pieces in k-fold cross-validation.

2.2.1 Metrics for classification

For the classification models we will use the area under the receiver operating characteristic curve (ROC AUC) and accuracy as our main metrics for model performance. Accuracy is defined as the fraction of correctly predicted observations:

$$\text{accuracy} = \frac{\text{TP + TN}}{\text{P+N}}$$

where TN and TP are the number of true positives and negatives, respectively, and P and N are the number of real positive and regative cases in the data, respectively.

We notice, by looking at Figure 2, that retained and non-retained small molecules are not equally present in our data. Due to this class imbalance accuracy alone is not a good metric, and hence we will consider other metrics as well. Sensitivity and specificity, which are also present in the ROC-curve, will be used and are defined as follows:

$$\text{sensitivity} = \frac{\text{TP}}{\text{P}}$$

and

$$\text{specificity} = \frac{\text{TN}}{\text{N}}.$$

We will also use the F1 score for assessing the final model. F1 score is defined as follows:

$$\text{F1 score} = \frac{\text{2TP}}{\text{2TP + FP + FN}},$$

where FP and FN are the number of false positives and negatives, respectively.

2.2.2 Metrics for regression

Regression models, used in predicting retention time, will be evaluated by using root mean squared error (RMSE), which is defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}$$

where y_i is the ith observation for the dependent variable, $\hat{y_i}$ is the corresponding prediction by the regression model, and n is the amount of observations.

Furthermore, we will use coefficient of determination (R^2) to assess how much of the variation seen in the dependent variable can be explained with the predictors. The coefficient of determination is defined in the following way:

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} = 1 - \frac{\sum_i (y_i - \hat{y_i})^2}{\sum_i (y_i - \bar{y})^2},$$

where RSS and TSS are the residual and total sum of squares, respectively, and $\bar{y}$ is the average value for the dependent variable.

The final regression model will also be evaluated by calculating the mean absolute error (MAE):

$$\text{MAE} = \frac{\sum_{i=1}^{n} |\hat{y_i} - y_i|}{n}.$$

MAE will help us compare model performance to the amount of experimental error observed in the retention time results.

3 Results & Discussion

Before diving into making predictions we will start this section by exploring the data. The results for our two main tasks are presented in separate sub-sections after exploratory data analysis.

3.1 Exploratory data analysis

3.1.1 Initial assessment of data

The dataset contains 80038 observations with 0 missing values. The 46 columns can be divided roughly into three categories: the dependent variable or the outcome (retention time), descriptive variables which can be used for identifying individual molecules, and the numerical features/predictors. The the first six values for the outcome variable along with the two ID variables are shown in Table 1.

The predictors can be further divided into subclasses based on the type of information they present. Rough division of predictors is as follows:

- Number of chemical elements in a specific small molecule are listed in the following columns: C, H, N, S, Cl, O, F, I, Br, Si, P
- Number of chemical groups are listed in: RNH2, R2NH, R3N, ROPO3, ROH, RCHO, RCOR, RCOOH, RCOOR, ROR, RCCH, RCN, RINGS, AROMATIC
- Values for physicochemical properties are listed in: MW, Ncharges, ALogP, ALogp2, AMR, apol, naAromAtom, nAromBond, TopoPSA, fragC, nHBAcc, nHBDon, nRotB, VABC, Zagreb, ECCEN, WPATH, WPOL

The distribution of chemical elements are displayed in Figure 4 in the form of a boxplot. Carbon and hydrogen atoms are present in abundance, which is not a surprize since we are dealing with organic molecules. The violin plots in Figure 4 b) show the distributions of carbom and hydrogen in more detail.

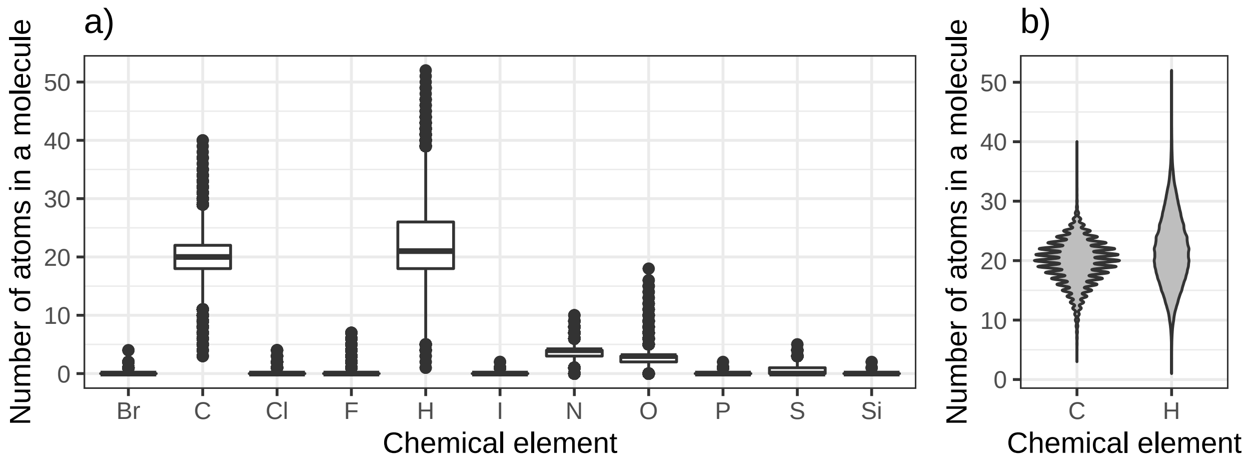

Figure 4: Boxplots of chemical element distributions (a) in the data are presented, along with violin plots (b) which show the distributions for carbon and hydrogen in greater detail.

Chemical group counts are shown in Figure 5. Many of the groups seem to be quite rare, whereas ring structures are commonly observed in the small molecules. The size of the rings in the dataset are limited to six carbon atoms or less, so if larger ring structures are present in a molecule they are not included in the count.

Table 1: The outcome variable and the two ID variables

Retention time (s)	PubChem ID	Molecular Formula
93.5	5139	$C_3H_8N_2S$
687.8	3505	$C_{19}H_{25}Cl_2N_3O_3$
590.7	2159	$C_{17}H_{27}N_3O_4S$
583.6	1340	$C_9H_7NO_2$
579.0	3344	$C_{15}H_{20}N_2O_2$
79.7	2277	$C_9H_{15}N_3O$

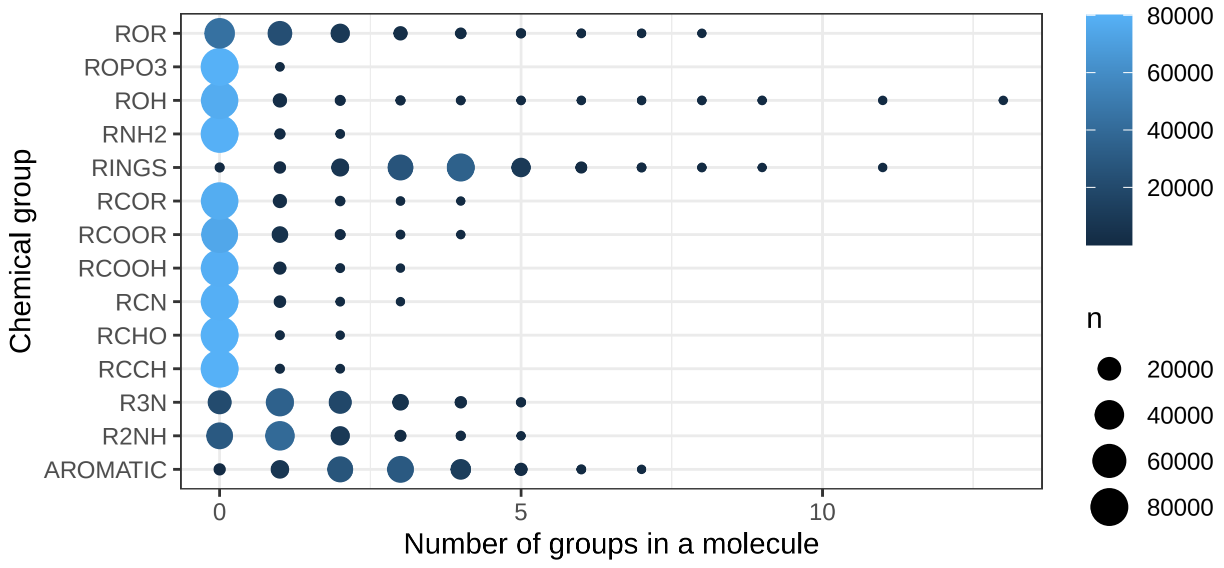

Figure 5: The appearance of different chemical groups in the dataset molecules are represented as a dotplot. The size and color of the dot are indicative of the amount observations. RINGS shows the amount of ring structures (up to six carbon atoms in size), and AROMATIC indicates how many of those rings are aromatic in nature.

Values for physicochemical properties are shown in Figure 6. Since the value for the properties take differing units it is difficult to present them on the same scale. Therefore, the variables are divided into two categories in Figure 6 based on the numeric property value to prevent the large-value variables from dwarfing the small-value ones. The variables with large numeric values represent the following molecular property descriptors:

- `WPATH`: the Wiener Path number (Wiener 1947)
- `fragC`: the calculated complexity of a system (Nilakantan et al. 2006)
- `ECCEN`: the eccentric connectivity index (Sharma, Goswami, and Madan 1997).

The smaller numeric value variable contain:

- `MW`: the molecular weight
- `ALogP`: the logarithm of the partition coefficient according to Ghose and Crippen (Ghose and Crippen 1986)
- `ALogp2`: the second power of `ALogP`
- `AMR`: the Ghose-Crippen molar refractivity (Ghose and Crippen 1987)
- `apol`: the sum of the atomic polarizabilities
- `TopoPSA`: the topological polar surface area based on fragment contributions (TPSA) (Ertl, Rohde, and Selzer 2000)
- `VABC`: the volume of a molecule
- `Zagreb`: a descriptor for the sum of the squared atom degrees of all heavy atoms
- `WPOL`: the Wiener Polarity number (Wiener 1947).

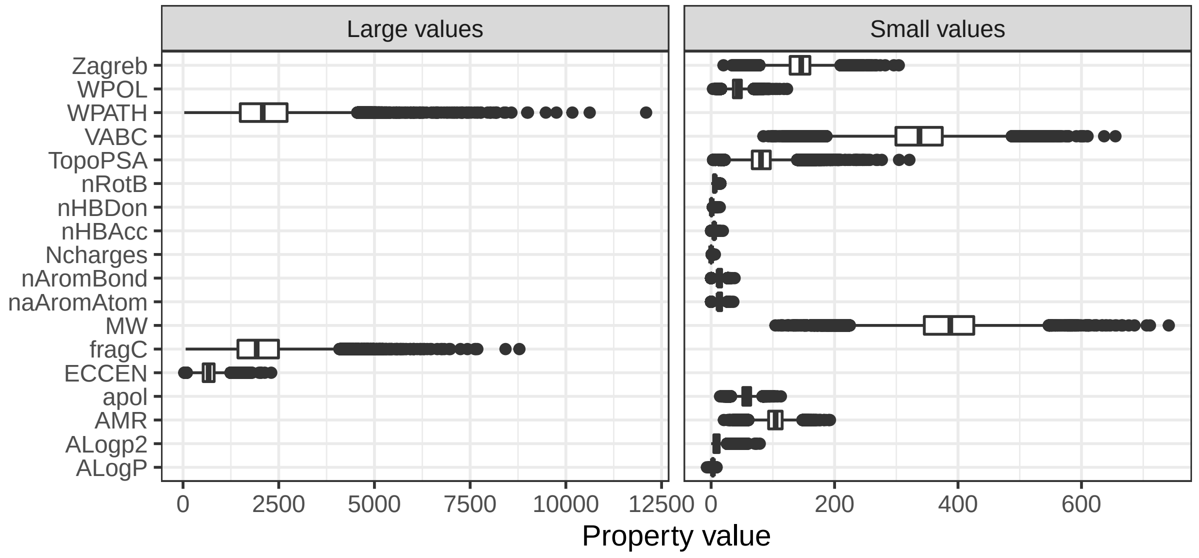

Figure 6: The distributions of physicochemical descriptors are shown in the form of boxplots. Variables with larger numeric values are shown on the left-hand side.

Additionally, some of the physicochemical descriptors, namely nRotB, nHBDon, nHBAcc, Ncharges, nAromBond, naAromAtom, are not continuous but rather counts, and are thus poorly displayed as a boxplot. These variables convey the following information:

- `nRotB`: the number of non-rotatable bonds on a molecule
- `nHBDon`: the number of hydrogen bond donors
- `nHBAcc`: the number of hydrogen bond acceptors
- `Ncharges`: the number of charged atoms
- `nAromBond`: the number of aromatic bonds
- `naAromAtom`: the number of aromatic atoms

Figure 7 shows the above mentioned variables in the form of histograms. Ncharges variable seems to have mainly zero-count observations, whereas the other variables in the figure are distributed more evenly.

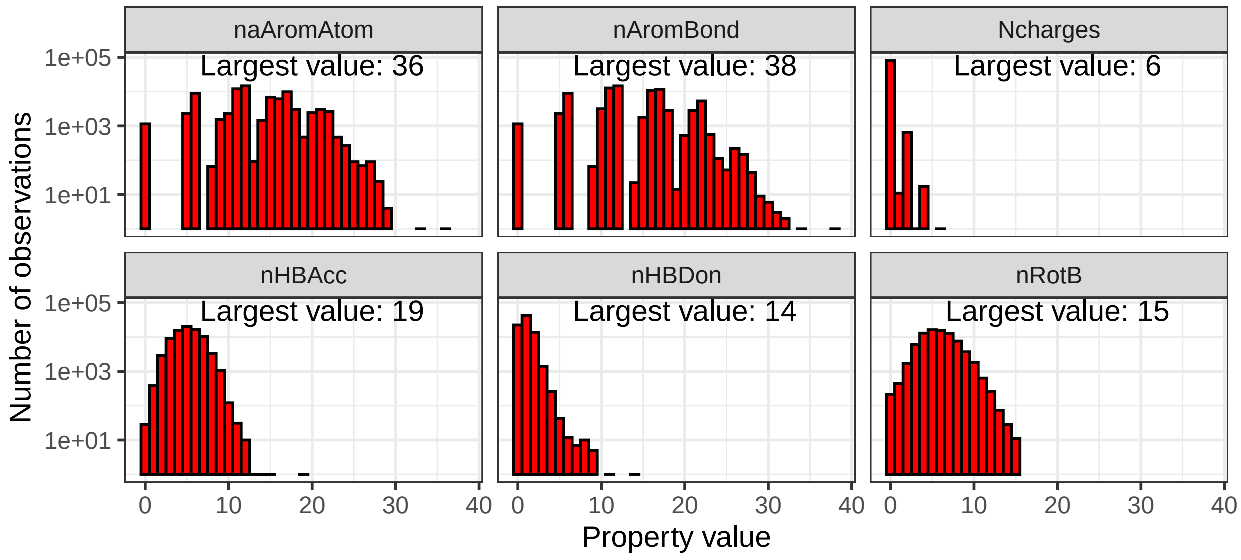

Figure 7: Values of certain physicochemical descriptors displayed as histograms. The largest observed value for each descriptor is depicted in the respective pane. Note the logarithmic y-axis.

3.1.2 Correlation between numerical data

All of the predictors are numerical variables, so let's examine their correlation. A visualization of the correlation matrix with the predictors and the outcome is shown in Figure 8. The variables at the bottom of the matrix seem to be highly correlated with each other.

Retention time has the highest correlation with ALogP with a value of 0.52. The highest negative correlation is with the variable R3N with a value of -0.29. For other variables, the absolute values of correlation with retention time remain below 0.5. Now that we have a general understanding on the structure of the data, let's briefly look at a few visualizations regarding the aims of this study.

3.1.3 Non-retained molecules

Earlier we defined molecules as non-retained, if they travel through the LC column is less than 300 seconds. This choice is somewhat arbitrary, and by no means a precise definition. However, by looking at Figure 2, we notice a clear gap between the retained and non-retained molecules, which supports our choice for the cut-off value between the classes.

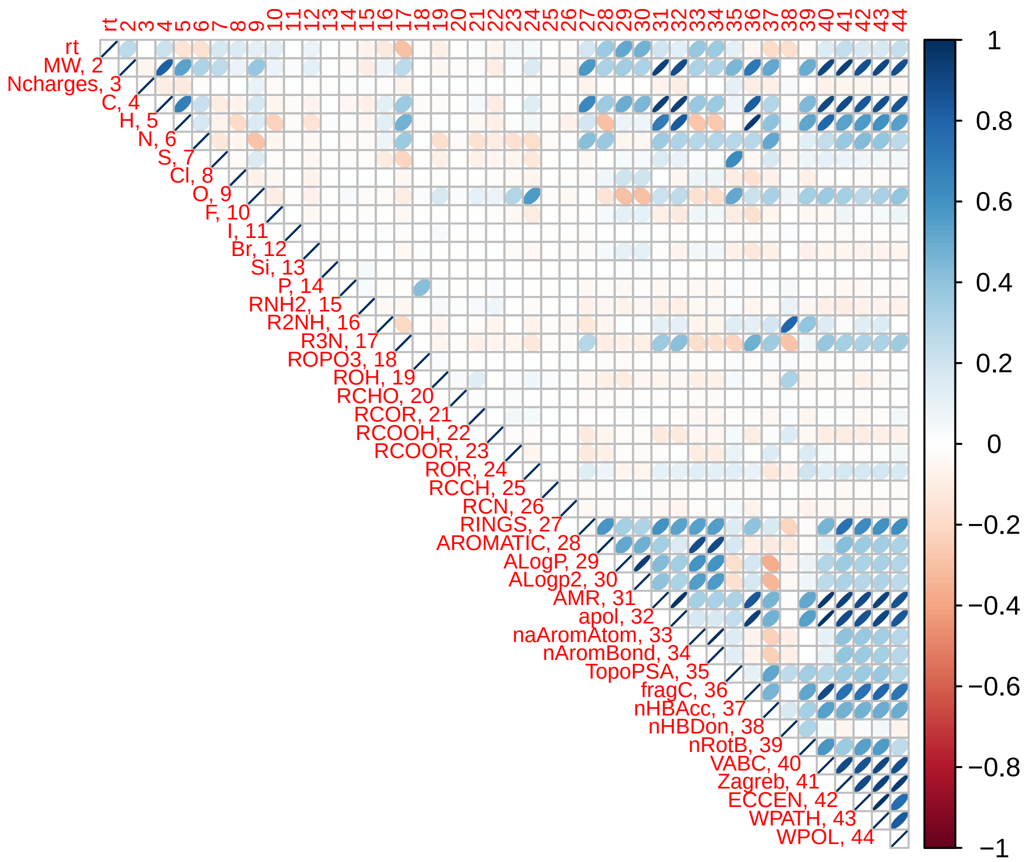

Figure 8: Visual representation of the upper half of the correlation matrix of retention time and the numerical predictors. The column names have been changed to column numbers to clarify the visualization. The column names and numbers are shown together as row names in the correlation matrix.

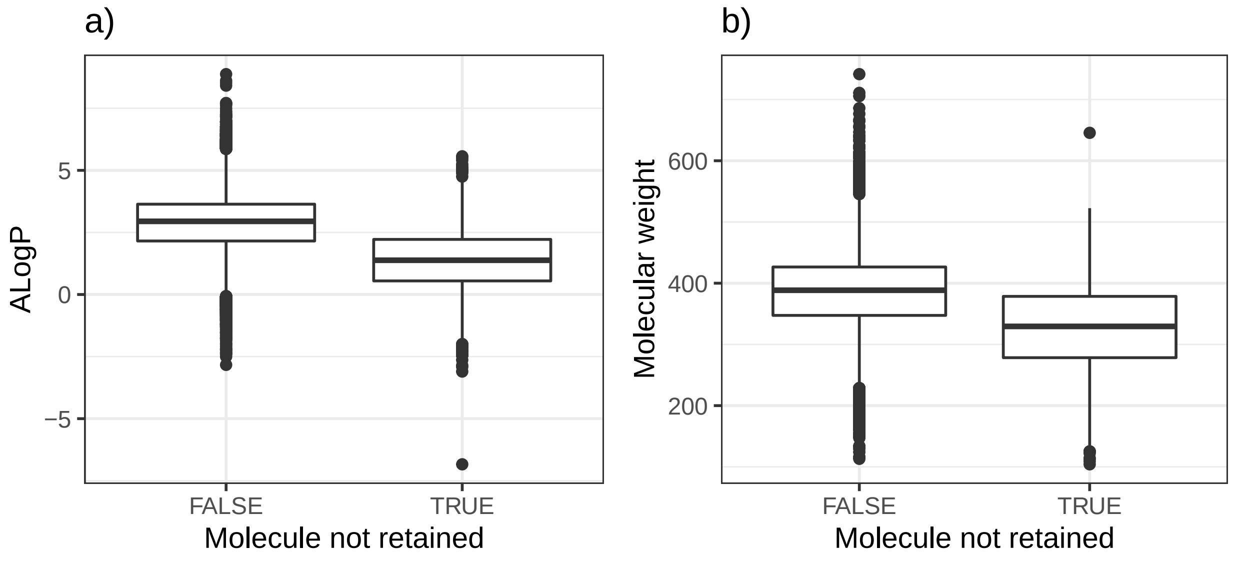

Figure 9: Boxplot of a) the partition coefficient and b) the molecular weight of retained and non-retained molecules.

Based on Figure 8, retention time had the highest correlation with the logarithm of the partition coefficient, `ALogP`. Moreover, we could postulate that smaller molecules travel through the LC column with relative ease. Figure 9 visualizes `ALogP` and molecular weight of retained and non-retained molecules. We notice that for both variables, the distributions of the two molecule classes are slightly shifted. This suggest that these variables might be useful in building a classification model. We will explore this topic further as we start building classification models.

Let's also look at principal component analysis (PCA) with the two classes of molecules. Figure 10 a) shows the biplot of the first two principal components, which reveals that even though the retained and non-retained molecules are somewhat separated, they are still mostly mixed. Furthermore, looking at Figure 10 b), we notice that the first two principal components only explain less than 40% of the variance. It would seem that using PCA will not provide any clear benefit for classification.

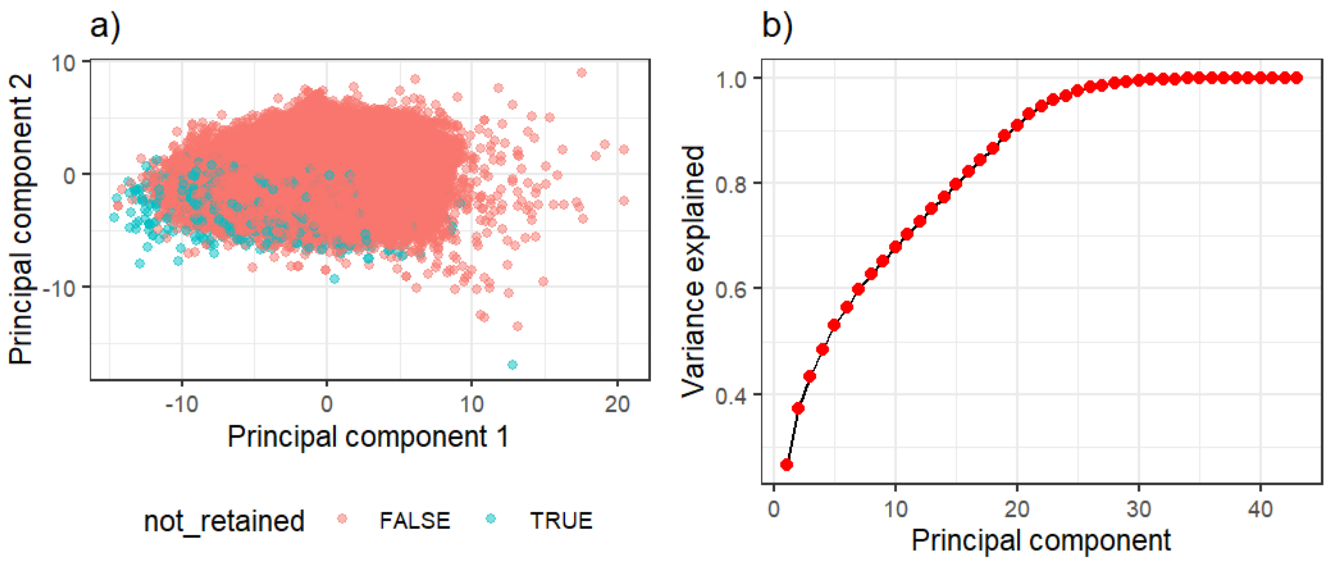

Figure 10: a) Biplot showing the first two principal components. The retained and non-retained molecules are coloured differently. b) Cumulative proportion of variance explained by the principal components.

3.1.4 Retention time

As we discussed earlier, `ALogP` has the largest correlation with retention time. Figure 11 visualizes relationship between `ALogP` and retention time. Non-retained molecules seem to lie as an uniform group at the bottom (i.e. near-zero correlation), whereas there seems to be a tendency for increasing retention time values as the `ALogP` values increase for retained molecules.

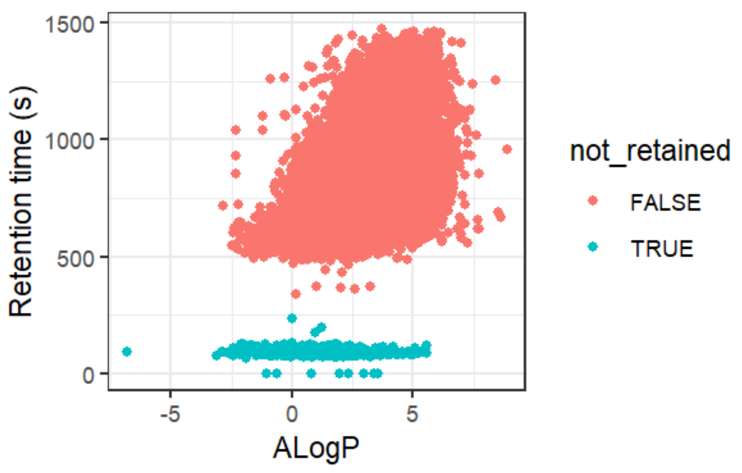

Figure 11: Retention time in seconds as a function of the logarithm of partition coefficient.

Table 2: The amount and proportion of observations in different classes for the test and train sets

not_retained	n	prop	set
FALSE	19491	0.974063	test
TRUE	519	0.025937	test
FALSE	58489	0.974362	train
TRUE	1539	0.025638	train

3.2 Classification of non-retained and retained molecules

In this section we will try different classification methods for evaluating whether a molecule is retained in the LC column or not.

Let's start by splitting the data to test and train sets. we will use 75% of the data for model building and comparison purposes (training set) and 25% for final model performance evaluation (test set). The amount of observations after the split are shown in Table 2. We can see that the data is quite imbalanced and retained molecules outnumber the non-retained ones. Nevertheless, the proportion of observations between the test and train sets are comparable, which is one of the things we paid attention to whilst making the data split. Looking at Table 2, our baseline accuracy, by assuming a model which categorizes every molecule as retained, is around 97.4 %. This is a tough benchmark to beat, but let's see how well our models fare.

We will test the following models for classification:

- Logistic regression
- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA)
- Random Forest.

The final model will be chosen by comparing model performance. We will use 10-fold cross-validation on the train set to assess the performance of several models. Due to class imbalance, other measures besides accuracy, such as specificity and area under the receiver operating characteristic curve (ROC AUC) will also be used. The final classification model performance will be evaluated against the test set using accuracy, sensitivity, specificity, and F1 score as evaluation metrics.

Table 3: Metrics for different logistic regression models were calculated based on 10-fold cross-validation

.metric	.estimator	mean	n	std_err	model	complexity
accuracy	binary	0.978	10	0.001	Logistic	all vars
roc_auc	binary	0.953	10	0.003	Logistic	all vars
accuracy	binary	0.975	10	0.000	Logistic	simple
roc_auc	binary	0.830	10	0.006	Logistic	simple
accuracy	binary	0.977	10	0.001	Logistic	intermediate
roc_auc	binary	0.944	10	0.003	Logistic	intermediate

3.2.1 Logistic regression

In logistic regression the probability of a molecule not being retained in the LC column (p_{nr}) is modelled using the logistic function:

$$p_{\text{nr}}(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}},$$

where β_0 and $\beta_1...\beta_p$ are the regression coefficients and $X = (X_1, ..., X_p)$ are p predictors (James et al. 2021). Fitting is used for determining values for individual regression coefficients, β_i.

We start by setting the values in `rt`, `pubchem` and `MF` columns as ID variables, which will not be used in fitting the model. The true classes are defined by a new column named `not_retained`, which is defined as being `TRUE` for molecules with `rt < 300`. Logistic regression models will be fitted with the remaining predictors.

First we fit a model with all variables as predictors. This leads to a pretty complicated model, so let's see if we can simplify our model by reducing the amount of predictors. The simplest model might contain only `ALogP` and `Mw` as predictors, since we saw in Figure 9 that these variables seem to be related to the retainability of a molecule.

An intermediate option would be to drop variables which we suspect not being related to the prediction outcome. Let's drop all chemical elements from our intermediate logistic regression model, namely the following columns: C, H, N, S, Cl, O, F, I, Br, Si, P. Furthermore, we will drop a few variables which are highly correlated with some of the other variables, namely ALogp2, apol, ECCEN, WPOL.

Estimated values for accuracy and ROC AUC based on 10-fold cross-validation for logistic regression models with different amount of variables are shown in Table 3. Accuracy is pretty similar for all models regardless of the amount of variables. However, ROC AUC is considerably improved with increasing model complexity. The intermediate model, which contains 15 variables less than the most complex model, performs nearly as well as the model which uses all predictors.

3.2.2 LDA, QDA & random forest

Logistic regression already gave results which were better than the baseline prediction accuracy of assuming all molecules as being retained. We can also try a few other supervised learning techniques for the classification task, and see if we can improve still. We will apply Linear and Quadratic Discriminant Analysis (LDA & QDA), and random forest to the classification task.

Both LDA and QDA are based on the Bayes' theorem, where the probability that observation Y belongs to a class k, given a certain predictor value $X = x$ is expressed by the following formula:

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)},$$

where K is the number of classes (in our case $K = 2$), π_k is the overall prior probability that a random observation belongs to the kth class, and $f_k(X)$ is the density function of X for an observation belonging to the kth class (James et al. 2021). Both LDA and QDA assume that the observations for each class are drawn from a multivariate normal distribution. The difference between LDA and QDA comes from the fact that LDA assumes that the K classes share a common covariance matrix, where as QDA estimates a separate covariance matrix for each class (James et al. 2021). The main practical difference resulting from this is that QDA is more flexible but also computationally more heavy. With p predictors, estimation of a covariance matrix requires estimating $p(p+1)/2$ parameters (James et al. 2021). For this reason, we will only test QDA with the so-called simple setting where the only predictors are `ALogP` and `MW`, and the intermediate setting, where the amount of predictors has been reduced by dropping the chemical elements and a few highly correlated variables. We used these same predictor selections earlier with logistic regression.

In a random forest model, we build several decision trees on bootstrapped training samples to estimate the class of an observation. Additionally, at each split we only use m out of the p predictors ($m < p$). This can be especially helpful if we have a large number of correlated predictors (James et al. 2021). Classification result for an individual observation will be the most common class predicted by the trees in the model.

After applying these new methods to our classification problem, random forest and logistic regression show the highest accuracy. Comparison of model performance in terms of accuracy is shown in Figure 12. Several models perform above the baseline accuracy, which comes from assuming all molecules as being retained in the LC column.

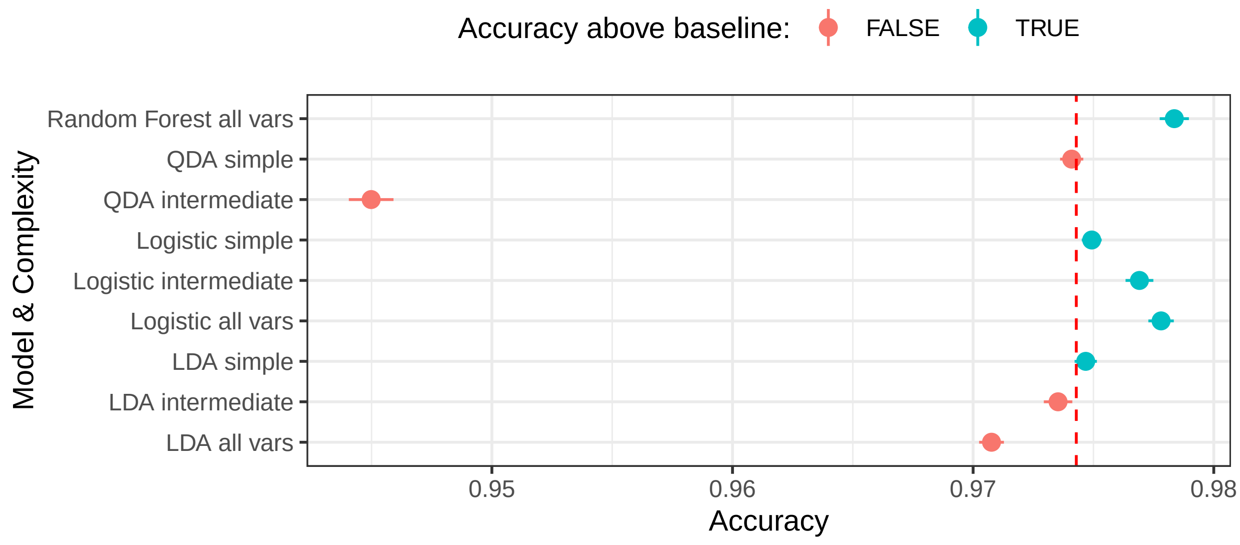

Figure 12: The mean accuracy of different models is shown in the form of a dotplot. The length of the error bars correspond with standard error defined through 10-fold cross-validation. The red dashed line indicates the baseline prediction accuracy which we aim to beat.

QDA does not seem to do well compared to the other models, whereas the performance of the simple version of LDA with just two predictors is quite comparable to the simple logistic regression.

ROC AUC was also assessed to get a better idea of model performance. Figure 13 shows the average ROC AUC values for the different models. Again, random forest and logistic regression with all the predictors perform the best.

The metrics for the two best performing models are presented in Table 4. We notice that performance is very similar for both models in terms of the metrics used here. Let's compare the performance of these two algorithms in more detail.

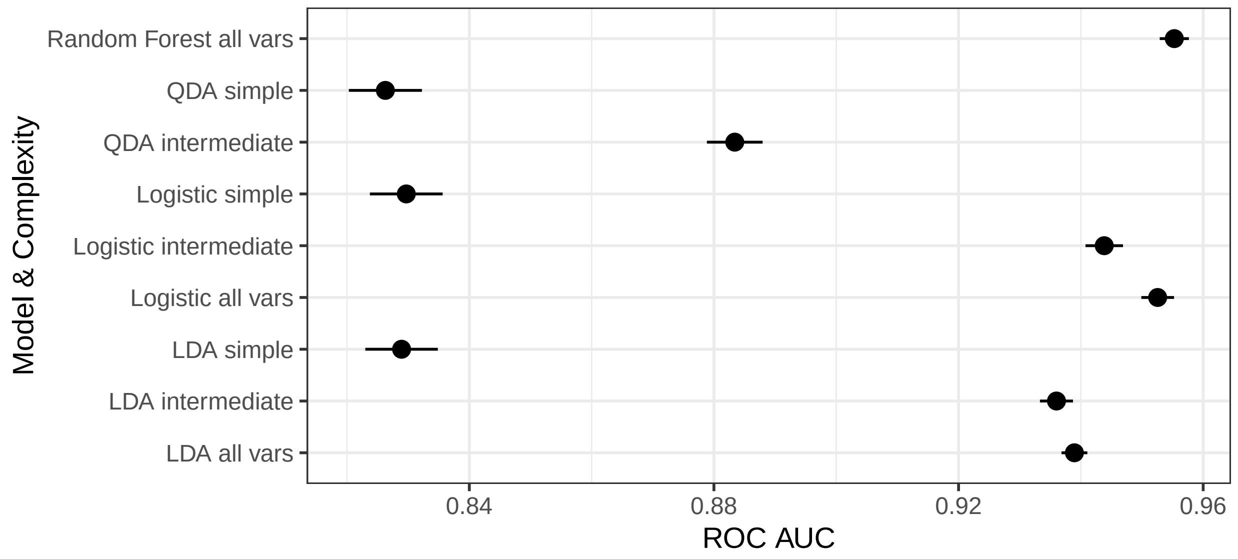

Figure 13: The mean ROC AUC values for different models is shown in the form of a dotplot. The length of the error bars correspond with standard error defined through 10-fold cross-validation.

Table 4: Accuracy and ROC AUC for the two best models

.metric	.estimator	mean	n	std_err	model	complexity
accuracy	binary	0.9778	10	0.0005	Logistic	all vars
roc_auc	binary	0.9526	10	0.0027	Logistic	all vars
accuracy	binary	0.9784	10	0.0006	Random Forest	all vars
roc_auc	binary	0.9553	10	0.0024	Random Forest	all vars

3.2.3 Choosing the final model

We will compare the performance of random forest and logistic regression using all predictors for training the models. 2-fold cross-validation will be used. That is to say, we split the train set into two pieces and assess model performace with these data sets. Training and validation of performance will be done twice, once per each half.

Figure 14 shows the ROC-curve comparison for both folds. The ROC-curves for both models are very similar, and both models seems to perform equally well. We could of course further fine tune model parameters in order to optimize model performance. For example, with random forest we could try to adjust the number of trees and the amount of predictors m used during each tree split. However, as our main objective lies in predicting retention time, we will not invest more time into parameter tuning at this point. Instead, we will choose the logistic regression as our final classification model, given the smaller computational resources required by this model. Next, we will assess the performance of logistic regression on the test set.

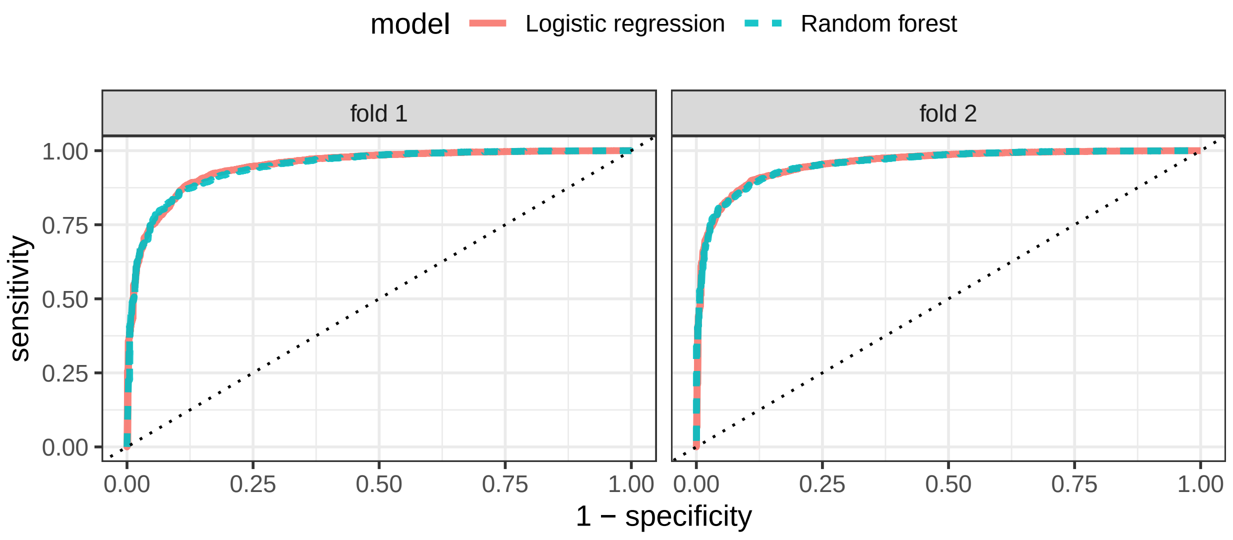

Figure 14: ROC curves for logistic regression and random forest models trained on the first and second fold of training data. The perfomance of the models is very similar.

3.2.4 Final classification model

Here we will fit our final classification model using the full training data set. Moreover, we will evaluate model performance with test data.

The terms with a p-value below 0.05 and estimated absolute regression coefficient value above one for the final model trained on the full training data set are shown in Table 5. `ALogP` and `apol` have the highest negative regression coefficient values, whereas the number of carboxylic acid groups `RCOOH` and the amount of esters `RCOOR` have the highest positive values.

Figure 15 shows the confusion matrix of final model prediction performance on test data as a heatmap. Retained molecules are properly classified is most cases. However, only 25.4 % of the non-retained molecules are correctly classified. Classification metrics for final model performance on test data are shown in Table 6. The model outperforms the baseline performance of the naive classifier model which would label all molecules as retained. However, the specificity of the final model is not very good.

Table 5: Regression coefficients for terms with a p-value below 0.05 and absolute estimated value above one for the final logistic regression model

term	estimate	std.error	statistic	p.value
Ncharges	1.268106	0.2094673	6.053956	0.0000000
RCOOH	3.022792	0.2273824	13.293867	0.0000000
RCOOR	2.406591	0.1959715	12.280311	0.0000000
RCN	1.343594	0.2509497	5.354035	0.0000001
ALogP	-2.074667	0.0810744	-25.589676	0.0000000
apol	-2.733214	1.3329035	-2.050572	0.0403087
nHBDon	1.242628	0.1149799	10.807348	0.0000000

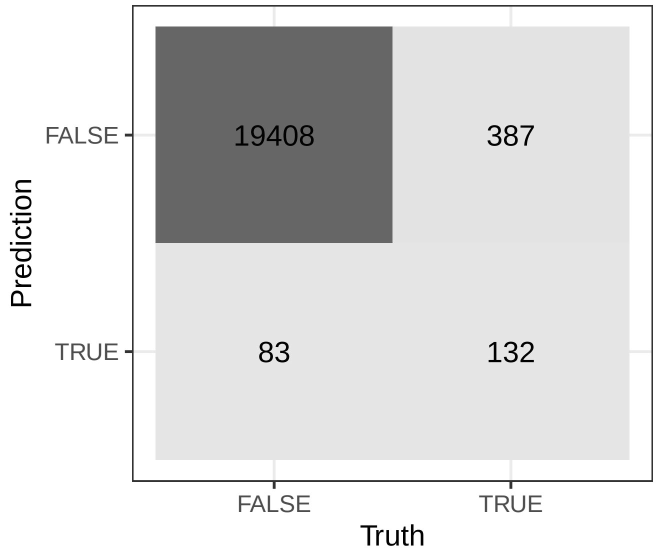

Figure 15: Confusion matrix showing the predicted and true classes for the dependent variable `not_retained`.

Table 6: Accuracy, sensitivity, specificity, and F1 score as metrics for the final classification model performance

.metric	.estimator	.estimate
accuracy	binary	0.9765117
sens	binary	0.9957416
spec	binary	0.2543353
f_meas	binary	0.9880365

3.3 Predicting retention time

In this section we will build models for predicting retention time. However, before making predictions, we should have an idea on the magnitude of experimental error included in the measured retention time values. All laboratory methods contain some level of intrinsic experimental error, and liquid chromatography is no exception. For example the LC column, which needs to be replaced regularly, has an effect on the measured values. When collecting the experimental results, the authors of the original study used a subset of 198 molecules to determine the variability of measurement results. Each of the 198 molecules was analyzed at least twice with a difference of at least 30 days. Variability was calculated for each individual molecule, and the observed mean and median variability for retention time was 36 and 18 s, respectively (Domingo-Almenara et al. 2019). This gives us some perspective when assessing the performance of regression models. For example, mean absolute error below 36 s on training data is indicative of overfitting.

On the opposite side of the error spectrum lies the RMSE we get by predicting the mean retention time of retained molecules as `rt` for all molecules. The RMSE value for this so-called null model is 174.7 s. This is our baseline prediction error, which we should be able to improve upon.

In order to develop and test different models, we again split our data in 25:75 proportions into test and training sets, respectively. The non-retained molecules with `rt` values below 300 s have been filtered out. Cross-validation will be used for assesing and comparing the performance of different models before final model validation on test data. Figure 16 shows that the distributions for retention time values are similar for both train and test sets.

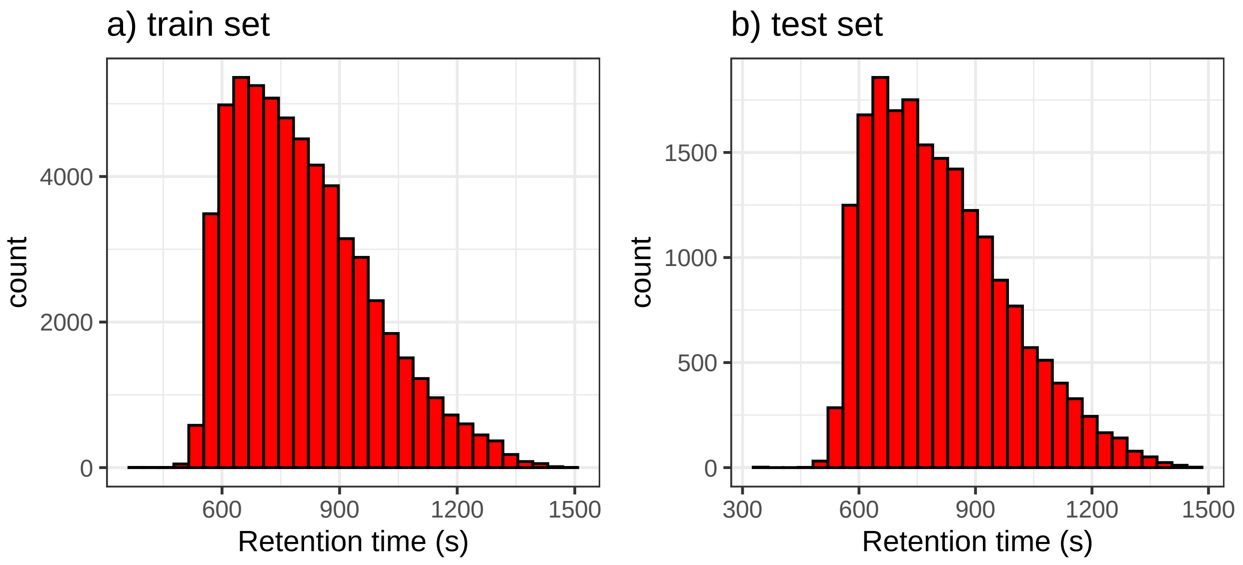

Figure 16: Distribution of retention time for train (a) and test (b) sets.

3.3.1 Linear regression

As a benchmark we will train a few linear regression models and see how well they do in predicting retention time. Let's begin with a simple model, where retention time is the expressed by the following formula:

$$\text{rt} = \alpha + \beta_1(\text{MW}) + \beta_2(\text{ALogP}) + \epsilon \tag{1}$$

The simple model shown above was evaluated using 10-fold cross-validation on the training data. The RMSE and R^2 values are shown in Table 7. The RMSE values are well above the mean experimental error of 36 s, but at the same time below the baseline RMSE of the null model (174.7 s).

Table 7: RMSE and R-squared metrics for the simple linear regression model

.metric	.estimator	mean	n	std_err
rmse	standard	150.3343	10	0.4456
rsq	standard	0.2599	10	0.0030

Table 8: RMSE and R-squared metrics for the linear regression models with more predictors

.metric	.estimator	mean	n	std_err	model
rmse	standard	122.1802	10	0.5136	intermediate
rsq	standard	0.5112	10	0.0034	intermediate
rmse	standard	120.7925	10	0.5000	all vars
rsq	standard	0.5222	10	0.0031	all vars

In order to improve model performance, we can try adding more predictors. The following variables Ncharges, I, Br, Si, P, RNH2, ROPO3, ROH, RCHO, RCOR, RCOOH, RCCH, RCN display low variance, and hence, might not add much information when trying to predict retention time. Let's fit two new models, one with all numerical features as predictors, and other with the low variance variables removed. Table 8 shows the metrics for these two models. We notice, that indeed the difference between the two models is not that large. Moreover, both models improve in terms of the metrics compared to the simpler linear model. Table 9 shows some of the statistically most significant terms in the linear model with all predictors fitted on the entire training set. The most significant predictors, in terms of p-value, are `ALogP`, `nHBDon`, and `R3N`.

3.3.2 Adding regularization to the model

While fitting the least squares estimates for the linear regression model, we did not observe signs of overfitting. However, it might be possible to drop some of the predictors and make the model easier to interpret. On the other hand, selecting the appropriate subset of predictors is not necessarily an easy task. Therefore, we will try to apply regularization to shrink the regression coefficients towards zero with a penalty parameter λ. We will use ridge regression and the lasso as shrinkage methods, and see if we can improve model performance.

In ridge regression, the regression coefficients are estimated by minimizing the following quantity:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\beta_j^2,$$

where the first term is the residual sum of squares (RSS), which we are minimizing when applying the least squares method for normal linear regression, and $\lambda \geq 0$ is a regularization parameter which is tuned (James et al. 2021). Figure 17 shows the RMSE and R-squared values for 10-fold cross-validation. We notice that as

Table 9: Model terms with the smallest p-values

term	estimate	std.error	statistic	p.value
(Intercept)	565.24880	11.2855089	50.08625	0
ALogP	82.78169	1.6825671	49.19964	0
nHBDon	-82.46758	1.8016211	-45.77410	0
R3N	-44.23220	0.9025427	-49.00843	0
ROR	-36.46596	1.1480738	-31.76273	0
naAromAtom	50.55463	1.6654484	30.35497	0

Table 10: RMSE and R-squared metrics for the best performing ridge regression model

penalty	.metric	.estimator	mean	n	std_err
1e-05	rmse	standard	123.44950	10	0.51286
1e-05	rsq	standard	0.50251	10	0.00355

the prediction error increases along with larger regularization parameter (λ) values. With the largest values the RMSE reaches the same level as the null model baseline prediction error.

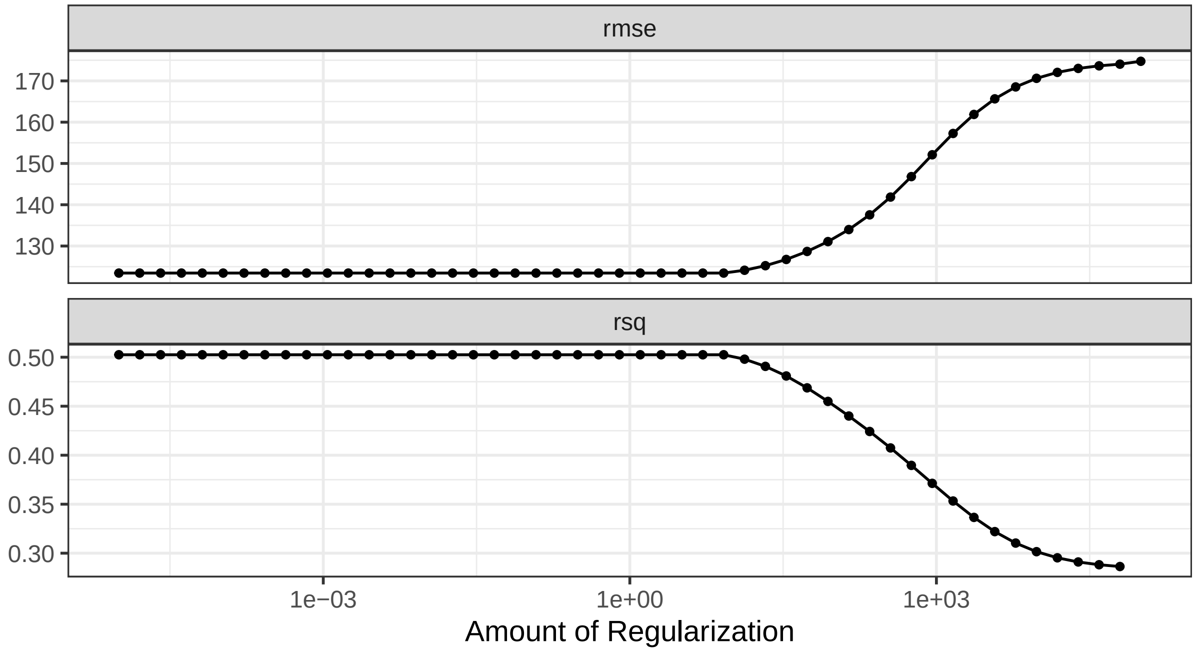

Figure 17: Rigde regression performance metrics calculated with 10-fold cross-validation for different values of the regularization parameter.

The best performance in terms of RMSE is obtained with the smallest λ-value. This indicates that the amount of regularization is the smallest possible, and the model is very close to the linear regression model with all variables. The results of 10-fold cross-validation for this model are shown in Table 10, and we notice that the performance is not improved in comparison to linear regression.

Lasso is an alternative to ridge regression. In lasso the quantity to minimize is as follows:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p} |\beta_j|,$$

where again the first term is the RSS, but the following regularization term at the end is different from ridge regression. Results for 10-fold cross-validation are shown in Figure 18. We see the same behavior as with ridge regression, where the best performance is obtained with the smallest amount of regularization. As the penalty term increases the coefficients shrink to zero and we essentially get the same level of performance as with the null model.

With lasso as well, the best performance in terms of RMSE is obtained with a small λ-value. The results of 10-fold cross-validation for the best lasso model are shown in Table 11. The performance is very close to linear regression, which is not a surprize considering the small value of the penalty parameter. Regularization

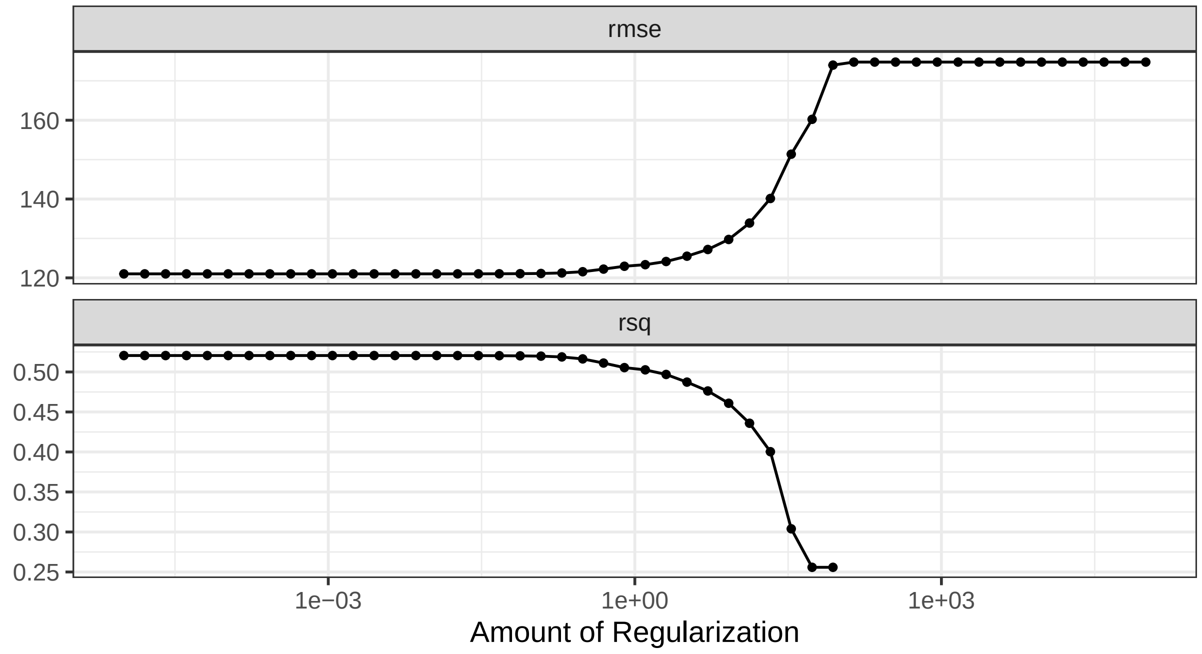

Figure 18: Lasso performance metrics calculated with 10-fold cross-validation for different values of the regularization parameter.

Table 11: RMSE and R-squared metrics for the best performing lasso model

penalty	.metric	.estimator	mean	n	std_err
0.0115	rmse	standard	121.0103	10	0.5018
0.0115	rsq	standard	0.5205	10	0.0033

does not seem to improve the prediction performance of the linear regression model. This makes sense as the amount of observations is large compared to the amount of predictors.

3.3.3 Polynomial regression

Regularization did not help us improve our model. Let's see if adding polynomial terms to the linear regression model helps us. We keep all variables and additionally raise the predictors related to physico-chemical properties (namely MW, Ncharges, ALogP, ALogp2, AMR, apol, naAromAtom, nAromBond, TopoPSA, fragC, nHBAcc, nHBDon, nRotB, VABC, Zagreb, ECCEN, WPATH, WPOL) to the second and third power. These new second and third order polynomial terms will be added to the model. Table 12 shows the metrics for this model determined through 10-fold cross validation. We notice that this helps us improve slightly on both metrics. However the improvement is not a massive one.

Table 12: RMSE and R-squared metrics for the cubic regression model

.metric	.estimator	mean	n	std_err
rmse	standard	118.786	10	0.4519
rsq	standard	0.538	10	0.0027

Table 13: RMSE and R-squared metrics for a regression tree model

.metric	.estimator	mean	n	std_err
rmse	standard	143.8272	10	0.4635
rsq	standard	0.3227	10	0.0023

Table 14: RMSE and R-squared metrics for the best performing regression tree model

cost_complexity	.metric	.estimator	mean	n	std_err
1e-04	rmse	standard	117.0749	10	0.532
1e-04	rsq	standard	0.5561	10	0.004

3.3.4 Regression trees

Above, we tested several linear regression models with different tweaks. Next we will see if tree-based methods could bring some benefits. We start with a simple regression tree model.

Table 13 shows the RMSE and R-squared for a regression tree model, which was evaluated on the training data using 10-fold cross-validation. The initial results in terms of the metrics is not competitive compared to the best models we created earlier.

We can try to improve model performance my tuning the cost complexity parameter. The tuning results are shown in Figure 19. The smallest parameter value seems to yield the best results.

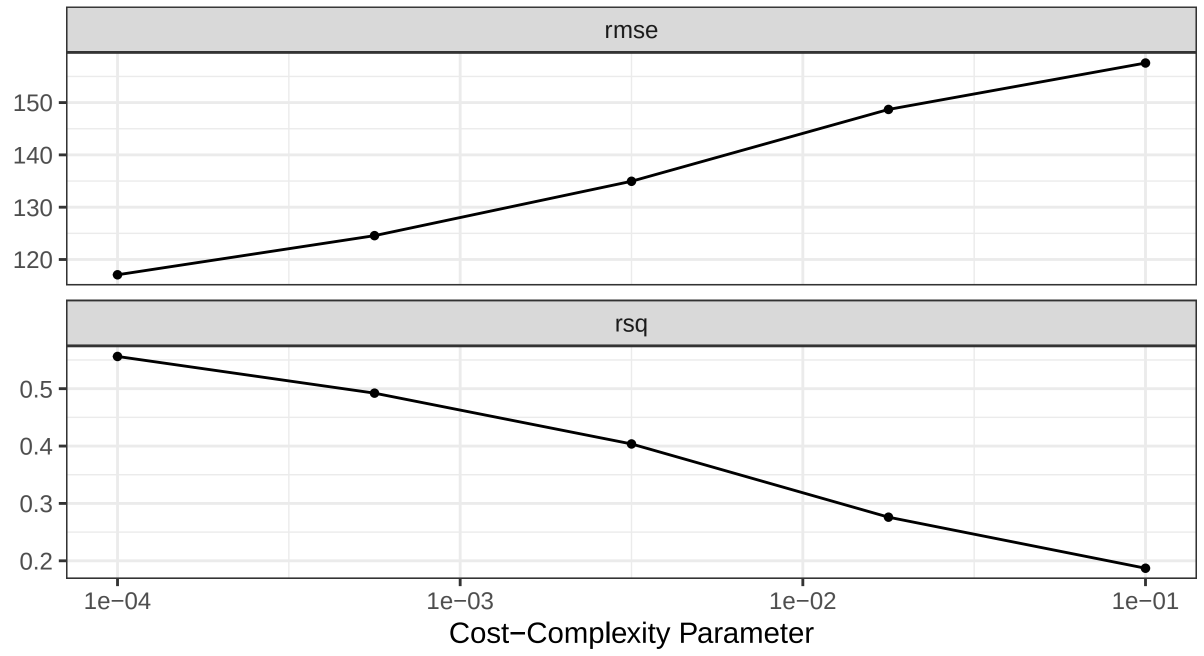

Figure 19: Tuning results for cost complexity parameter.

The performance metrics for the model with the best cost complexity parameter are shown in Table 14. By tuning the cost complexity parameter we were able to get the best prediction results thus far.

The downside of fitting a regression tree on a dataset with this many predictors is that the tree structure becomes so large and complex that visualization of all splits is not very helpful anymore. However, what we can visualize are the most important variables. This in fact will help us increase our understanding regarding which features affect the `rt` the most.

For visualizing the variable importance, we first train the best performing tree model again with the entire training data. The most important variables are now shown in Figure 20. Again we see that `ALogP` is connected to retention time. Aromatic rings, along with a few other variables, seem to be important as well.

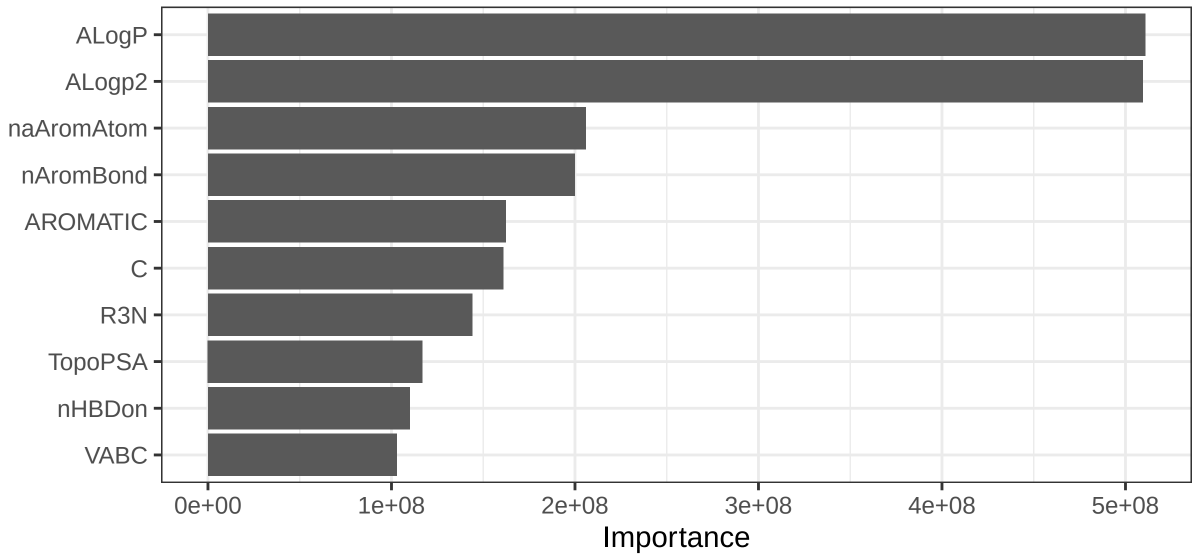

Figure 20: Most important variables in the regression tree model.

Although our tree model above outperformed the other models, there are a few disadvantages in using such a model. One issue concerning tree-based models is the fact that they can be very non-robust. To rephrase, small changes in data can lead to large changes in the final tree structure (James et al. 2021). There are several methods to overcome this problem. Here we will apply boosting.

In boosting we first fit a very small tree to the data. In some cases the tree size can be as small as a single split. Next step is to fit another small tree to the residuals. This new tree will be added to the previous tree to improve model fit. Additionally a shrinkage parameter is applied to the trees to slow down the learning rate. After recursively fitting several small trees to the updated residuals, the final model can be expressed with the following equation:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x),$$

where B is the total amount of small trees in the model, λ is the shrinkage parameter, and $\hat{f}^b(x)$ is the bth small tree in the model.

Let's fit a boosted tree model with 1000 trees which have a tree depth of two splits. The results of 10-fold cross-validation are shown in Table 15. The boosted tree model outperforms all other models by a clear margin. Let's try to tune the model parameters to further improve the performance.

3.3.5 Tuning the boosted tree model

The boosted tree model contains many parameters for us to tune, such as `tree_depth`, `learn_rate`, and `trees` among a few other parameters. This makes a standard grid search very inefficient and time consuming if we try to tune all possible parameters. There are different ways for tackling this problem. For example, the

Table 15: RMSE and R-squared metrics for a boosted tree model containing 1000 trees

.metric	.estimator	mean	n	std_err
rmse	standard	106.4804	10	0.4941
rsq	standard	0.6289	10	0.0030

Table 16: RMSE for the two best models found during parameter tuning

trees	tree_depth	.metric	mean	n	std_err
1500	5	rmse	96.1512	10	0.5277
1000	5	rmse	96.4256	10	0.5222

finetune package contains functions such as `tune_race_anova()`, which terminates training of non-promising parameter combinations early (Kuhn and Silge 2022). This can lead to significantly shorter execution times for the code. Here we will however restrict ourselves to tuning two of the parameters, namely `trees` and `tree_depth`, in a more traditional fashion.

As our first step for tuning, we set the tuning grid values for `tree_depth` as 1, 2, 3, 4, and for `trees` as 500, 1000, 1500. The RMSE of models trained with these different parameter values show that 500 trees do not guarantee sufficient performance compared to larger models. We also notice that increasing `tree_depth` might further improve the results. Therefore, we add two more parameter combinations to our grid search, namely 1000 and 1500 `trees` with a `tree_depth` of 5. The tuning results via 10-fold cross-validation are shown in Figure 21. The best RMSE results are obtained with a `tree_depth` value of 5.

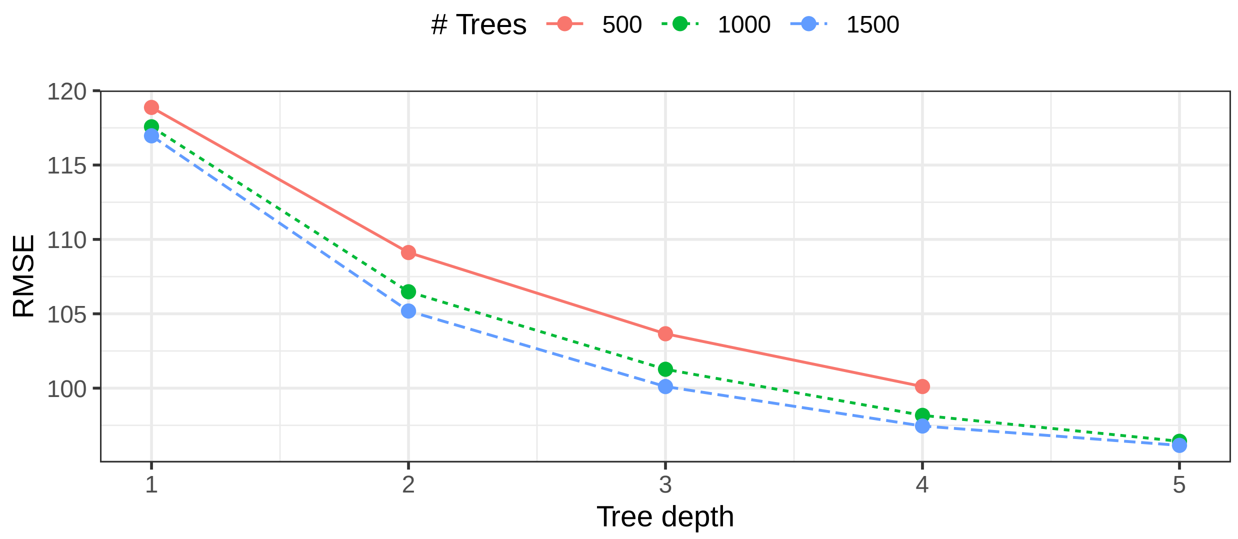

Figure 21: Tuning results for different number of trees and tree depth in the model. RMSE was calculated using 10-fold cross-validation.

The two best perfoming models are shown in Table 16. The RMSE for the models with 1500 and 1000 trees is quite similar. Therefore, we will pick the simpler model with 1000 trees and tree depth of 5 as our final model.

Table 17: RMSE, R-squared, and MAE of the final model evaluated on the test data

.metric	.estimator	.estimate
rmse	standard	107.5482
rsq	standard	0.6204
mae	standard	79.0322

Table 18: Number and proportion of predictions within and outside 36 s of the true value

good_pred	n	prop
FALSE	13064	0.67
TRUE	6431	0.33

3.3.6 Training and evaluating the final model

Let's train our final regression model using the entire training data. We will evaluate model performance on the test data. The results of the final model performance on the test set are shown in Table 17.

The model performs reasonably well on the test data. However, we are still not at the limit set by the mean experimental error of 36 seconds. This can be seen better as we visualize the residuals. Figure 22 shows that the residuals are evenly distributed around zero. Moreover, a large portion of the predictions are within the experimental error, although some predictions are quite far from the true value.

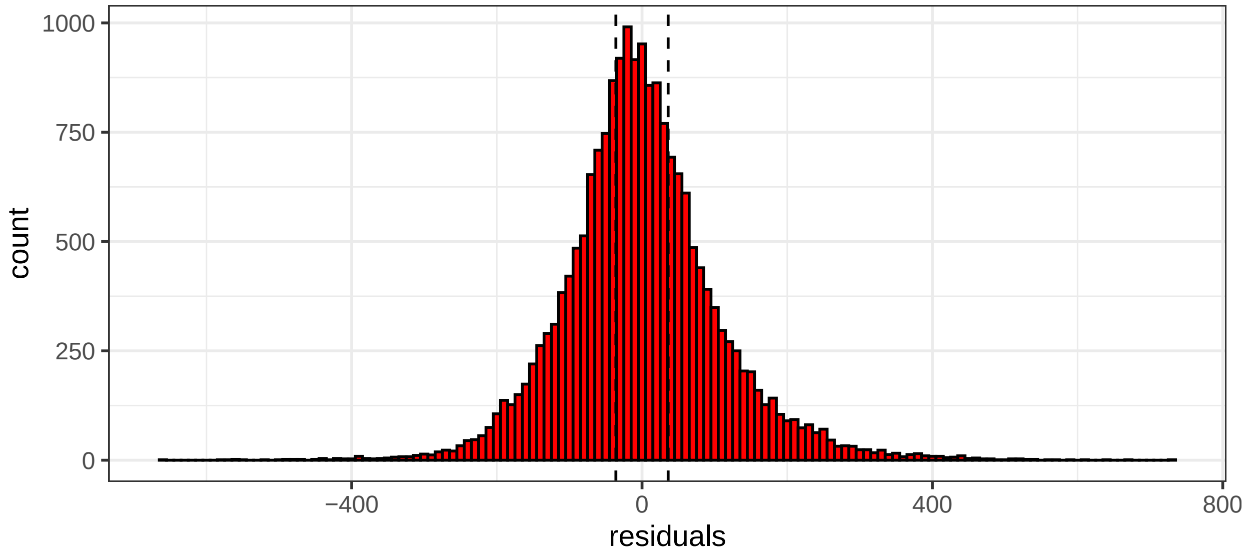

Figure 22: Residuals calculated for the final model are evenly distributed. The vertical dashed lines show the average experimental error of 36 s, determined in the original study.

Table 18 shows the number and proportion of predictions with an absolute error smaller than 36 seconds. We see that around third of the predictions are within the experimental error limits.

Finally, Figure 23 visualizes the true `rt` values versus the predictions. We see that in general the observations stack around the line depicting perfect prediction. Additionally, the predictions seem to hold quite well for large and small values of `rt` alike.

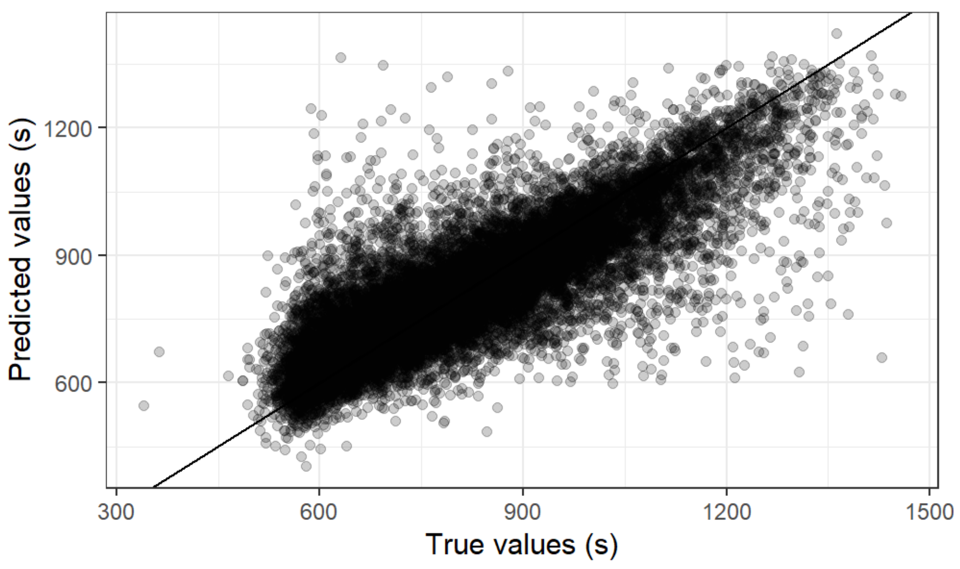

Figure 23: True retention time values versus the predicted values show fairly good correlation.

4 Conclusions

Our initial aim was to classify small molecules into non-retained and retained groups. Moreover, we wanted to predict the retention time for the retained molecules passing through the LC column using a simple set of molecular descriptors.

Logistic regression and random forest models were most effective in the classification task. Even though the large imbalance between the classes made the task difficult, our final model was able to beat the baseline performance of the naive classifier model. The logarithm of the partition coefficient `ALogP`, the sum of the atomic polarizabilities `apol`, along with the amount of carboxylic acid groups and esters were identified as important variables for classification.

The basic linear regression model faired quite well in predicting retention time. However, in the end we were able to improve model performance by using a boosted tree model. `ALogP` popped up several times when important variables for different models were assessed. The final model was reasonably good at predicting retention time, but there is clearly still room for improvement. More detailed tuning of the boosted tree model would most likely improve model performance even further. However, it is quite probable that the set of predictors we used for training our models does not contain all essential molecular information needed for accurately predicting retention time. Better model performance would no doubt be obtained by using more sophisticated molecular fingerprints as predictors.

In summary, it was shown that simple molecular descriptors can be used for building fairly good models for retention time prediction.

References

Cao, Dong-Sheng, Nan Xiao, Qing-Song Xu, and Alex F. Chen. 2014. "Rcpi: R/Bioconductor package to generate various descriptors of proteins, compounds and their interactions." *Bioinformatics* 31 (2): 279–81. https://doi.org/10.1093/bioinformatics/btu624.

Cao, Y., A. Charisi, L. C. Cheng, T. Jiang, and T. Girke. 2008. "ChemmineR: a compound mining framework for R." *Bioinformatics* 24: 1733–34. https://doi.org/10.1093/bioinformatics/btn307.

Domingo-Almenara, X. 2019. "The METLIN Small Molecule Dataset for Machine Learning-Based Retention Time Prediction. (Dataset)." *Figshare* V1. https://doi.org/10.6084/m9.figshare.8038913.v1.

Domingo-Almenara, X., C. Guijas, E. Billings, J. R. Montenegro-Burke, W. Uritboonthai, A. R. Aisporna, E. Chen, H. P. Benton, and G. Siuzdak. 2019. "The METLIN Small Molecule Dataset for Machine Learning-Based Retention Time Prediction." *Nat. Commun.* 10: 5811. https://doi.org/10.1038/s41467-019-13680-7.

Ertl, P., B. Rohde, and P. Selzer. 2000. "Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties." *Journal of Medicinal Chemistry* 43 (20): 3714–17.

Ghose, A. K., and G. Crippen. 1987. "Atomic physicochemical parameters for three-dimensional-structure-directed quantitative structure-activity relationships. 2. Modeling dispersive and hydrophobic interactions." *Journal of Chemical Information and Computer Science* 27: 21–35.

Ghose, A. K., and G. M. Crippen. 1986. "Atomic physicochemical parameters for three-dimensional structure-directed quantitative structure-activity relationships. I. Partition coefficients as a measure of hydrophobicity." *Journal of Computational Chemistry* 7: 565–77.

James, G., D. Witten, T. Hastie, and R. Tibshirani. 2021. *An Introduction to Statistical Learning with Applications in R.* New York: Springer. https://doi.org/10.1007/978-1-0716-1418-1_1.

Kuhn, Max, and Julia Silge. 2022. *Tidy Modeling with R.* www.tmwr.org.

Nilakantan, R, D S Nunn, L Greenblatt, G Walker, K Haraki, and D Mobilio. 2006. "A family of ring system-based structural fragments for use in structure-activity studies: database mining and recursive partitioning." *Journal of Chemical Information and Modeling* 46: 1069–77.

Sharma, V, R Goswami, and A K Madan. 1997. "Eccentric Connectivity Index: A Novel Highly Discriminating Topological Descriptor for Structure-Property and Structure-Activity Studies." *Journal of Chemical Information and Computer Sciences* 37: 273–82.

Wiener, H. 1947. "Structural determination of paraffin boiling points." *Journal of the American Chemical Society* 69 (1): 17–20.