Curve Classification User Guide

Florian Seefried (florian.seefried@tum.de)

`https://github.com/kusterlab/curveClassification_shiny`
September 7, 2018

Contents

1 General Information

General information section explains general terms and the purpose for which the CurveClassification tool is intended.

1.1 Software overview

CurveClassification is a shiny application, which allows the training and application of a classification algorithm. The application provides a user interface to perform supervised classification. However, this application is adapted specifically to the requirements of classification problems occurring in biochemical binding and activity assays. The software is written in the language R and is accessible on github under `https://github.com/kusterlab/curveClassification_shiny` and could be run locally using R.

1.2 Organization of the Manual

The user's manual consists of five sections: General information, Software Summary, Getting Started, Using the Software and Video tutorial.

General information section explains general terms and the purpose for which the CurveClassification tool is intended.

Software Summary section provides a general overview of the software. The summary outlines the user's access levels and the corresponding limitations or requirements according to the access level.

Getting Started section explains how to access CurveClassification. Furthermore, the software menu is briefly explained.

Using the Software section provides a detailed description of software functions.

The Video tutorial part provides a embedded video that shows with an example dataset, how to use this tool.

2 Software Summary

Software Summary section provides a general overview of the software. The summary outlines the user's access levels and the corresponding limitations or requirements according to the access level.

2.1 User Access Levels

Everyone can access the code of the application under `https://github.com/kusterlab/curveClassification_shiny` to implement the application on local user computers or severs. The versions of the packages that were used to build the software are specified in the packrat file. Additionally, to the shiny application the package `https://github.com/kusterlab/curveClassification_package` is required.

2.2 Notices

CurveClassification is open source software and comes with absolutely no warranty.

3 Getting Started

Getting Started section explains how to access CurveClassification. Furthermore, the software menu is briefly explained.

3.1 Accessing the Software

The newest version currently available can be accessed under `https://github.com/kusterlab/curveClassification_shiny`.

3.2 Software Menu

CurveClassification is a tabbed application, which consists of 7 tabs (figure 1). Five tabs are intended to generate a new classifier. One tab enables the optimization of an existing model. The last tab enables a prediction for new data with a already existing model. The names of the tabs represents the action that can be performed under the respective tab. Furthermore, the tabs are sorted in a order (from left to right) that represents the logic order of actions that need to be performed in order to generate a new classifier. However, not for all tabs actions need to be performed to generate a model, the functionality of some tabs is only optional.

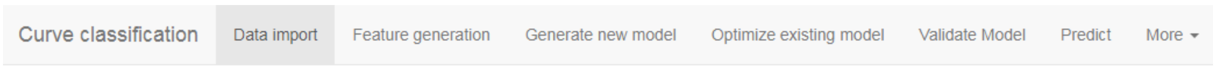
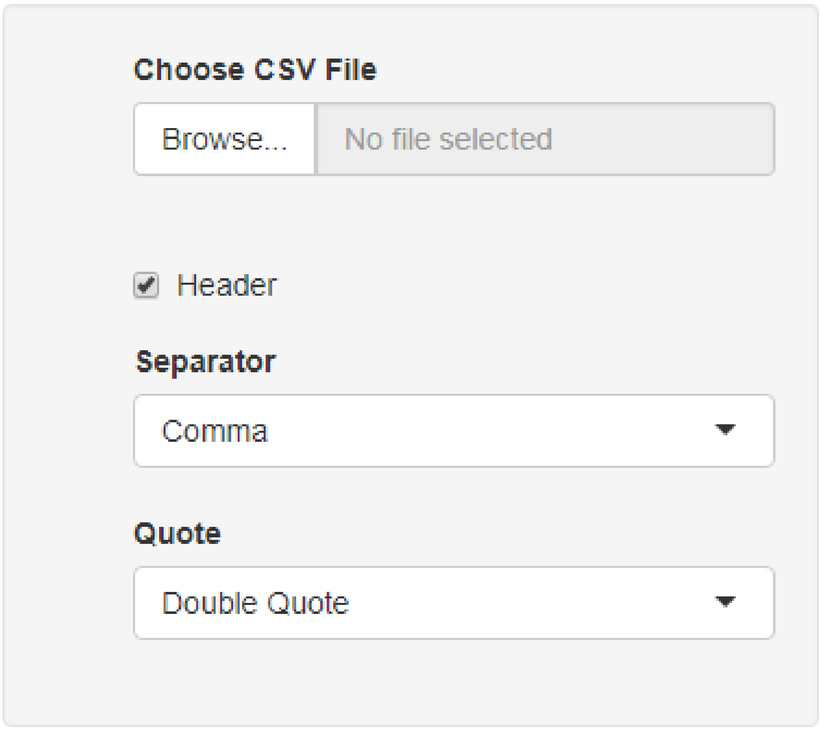


Figure 1: Main menu of the Curve Classification shiny application

3.2.1 Import data tab

The Import data tab (figure 2) consists of four fields. The first is a upload panel into which the required data sets could be uploaded. The checkbox *Header* can specify whether the uploaded data has a header or not. *Seperator* and *Quote* specify the properties of *.csv* that is uploaded.



Choose CSV File

Browse... No file selected

[✓] Header

Separator

Comma

Quote

Double Quote

Figure 2: Sub menu for the tab Import data

3.2.2 Calculate features tab

The Calculate features tab handles the feature generation functionality. It is an advanced feature that is not required for the initial model generation and allows the upload of functions that calculate new variables from the uploaded dataset. The basic idea is to provide additional information for the learning algorithm to make it easier to distinguish the two groups (*e.g.* non-linear curve fit). After calculation of those features the distributions of all features can be visualized between the positive and the negative group.

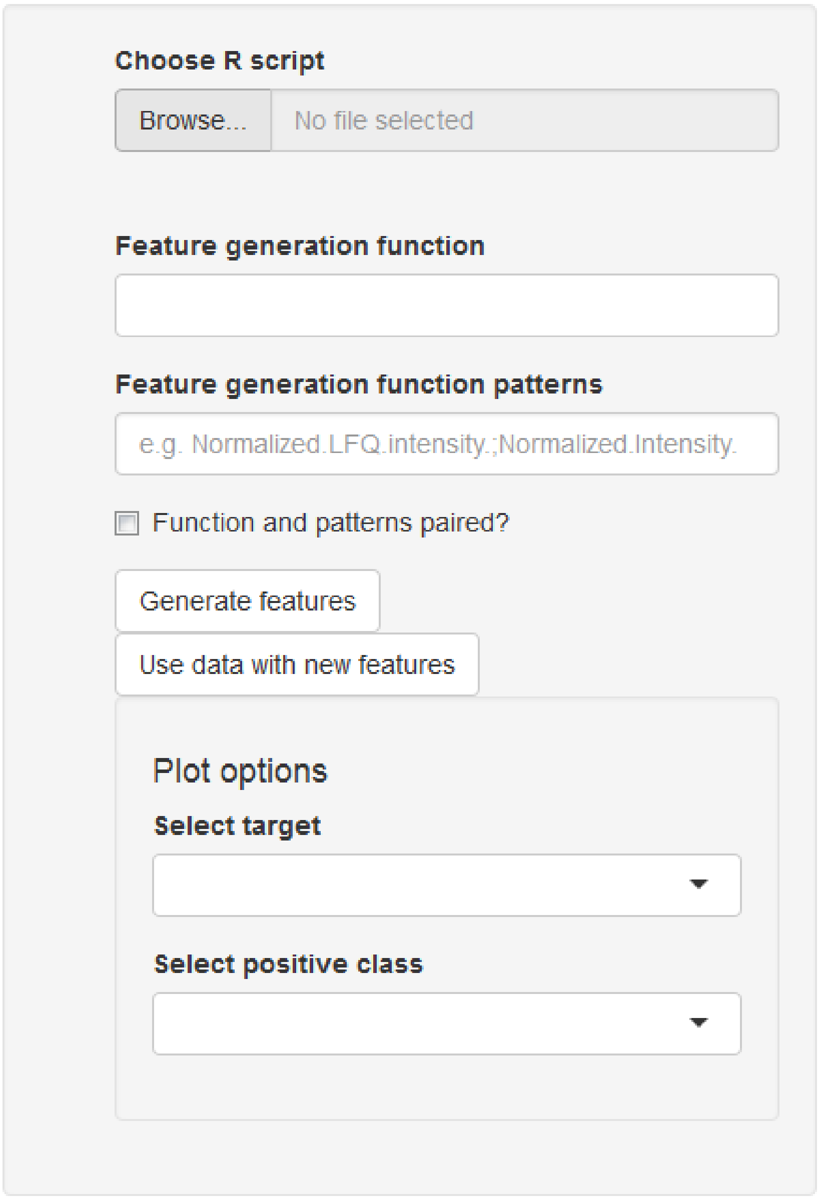

Choose R script
Browse...
No file selected
Feature generation function
Feature generation function patterns
e.g. Normalized.LFQ.intensity.;Normalized.Intensity.
Function and patterns paired?
Generate features
Use data with new features
Plot options
Select target
Select positive class

Figure 3: Sub menu for the tab Calculate features

3.2.3 Generate new model tab

The Generate new model tab manages the training of a new model based on the uploaded and conceivably modified data. All features that were not needed for the classification (do not contain any informative value) can be excluded under *Exclude features*. Subsequently, the target variable has to be selected and the positive class needs to be assigned. Afterwards a initial model can be trained and downloaded. All other inputs are only for advanced tuning options in latter steps of model generation. After a successful training the model performance is displayed in the main panel.

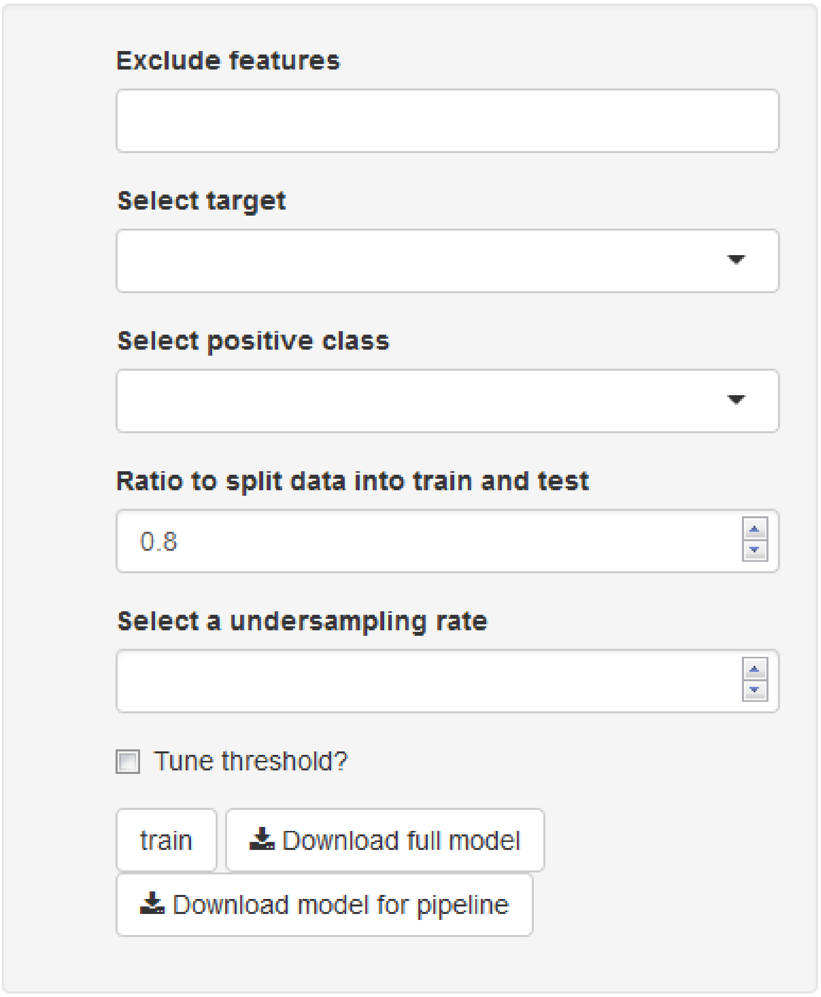

Exclude features
Select target
Select positive class
Ratio to split data into train and test
0.8
Select a undersampling rate
Tune threshold?
train
Download full model
Download model for pipeline

Figure 4: Sub menu for the tab Generate new model

3.2.4 Optimize existing model

The Optimize existing model tab handles the optimization of a existing model. The first used case is that a existing model can be redefined with additional manually annotated data. The second use case is the replacement of the initially used dataset with the same dataset which is manually re-annotated. The advantage of using this option in comparison to simply train a new model is that all parameters and feature calculation functions are kept. The performances of the new and the old model were compared in the main panel.

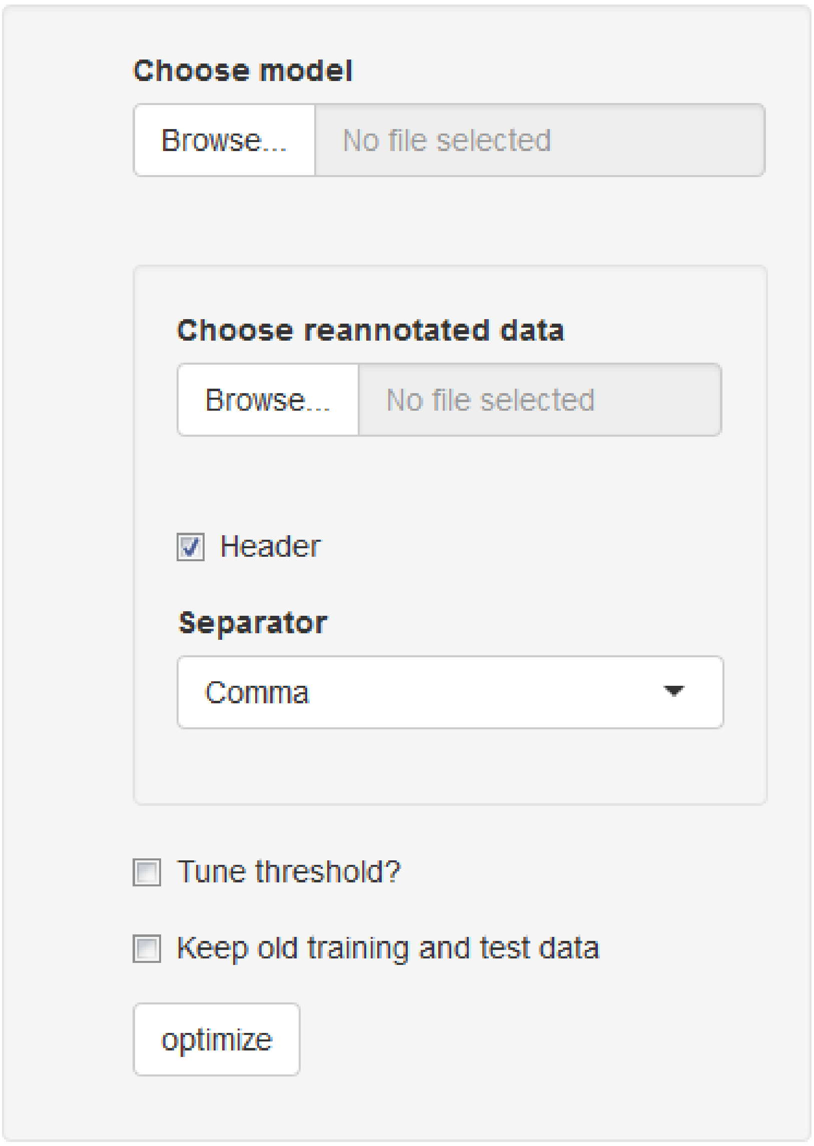


Figure 5: Sub menu for the tab Optimize existing model

3.2.5 Validate model

The Validate model tab visualizes the false positives and false negatives for the initial or optimized models, if a plot function is available. This information can be used to get a feeling what the model has learned and if the false predicted observations make sense.

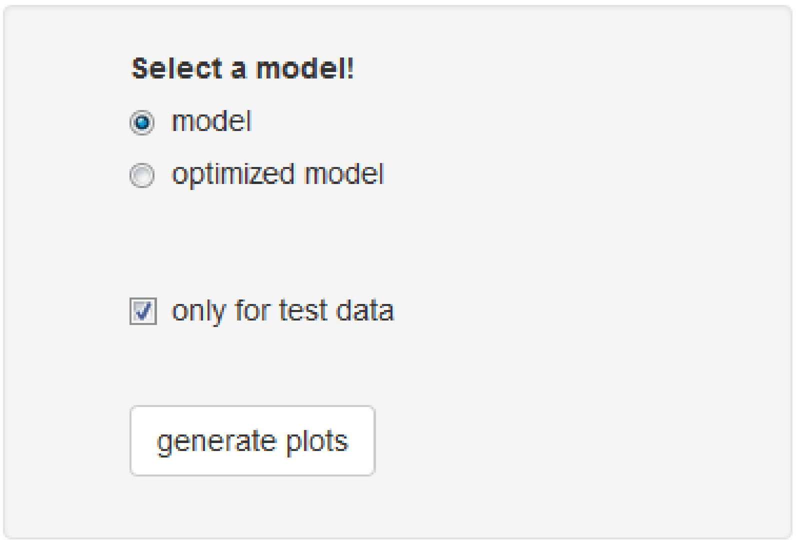
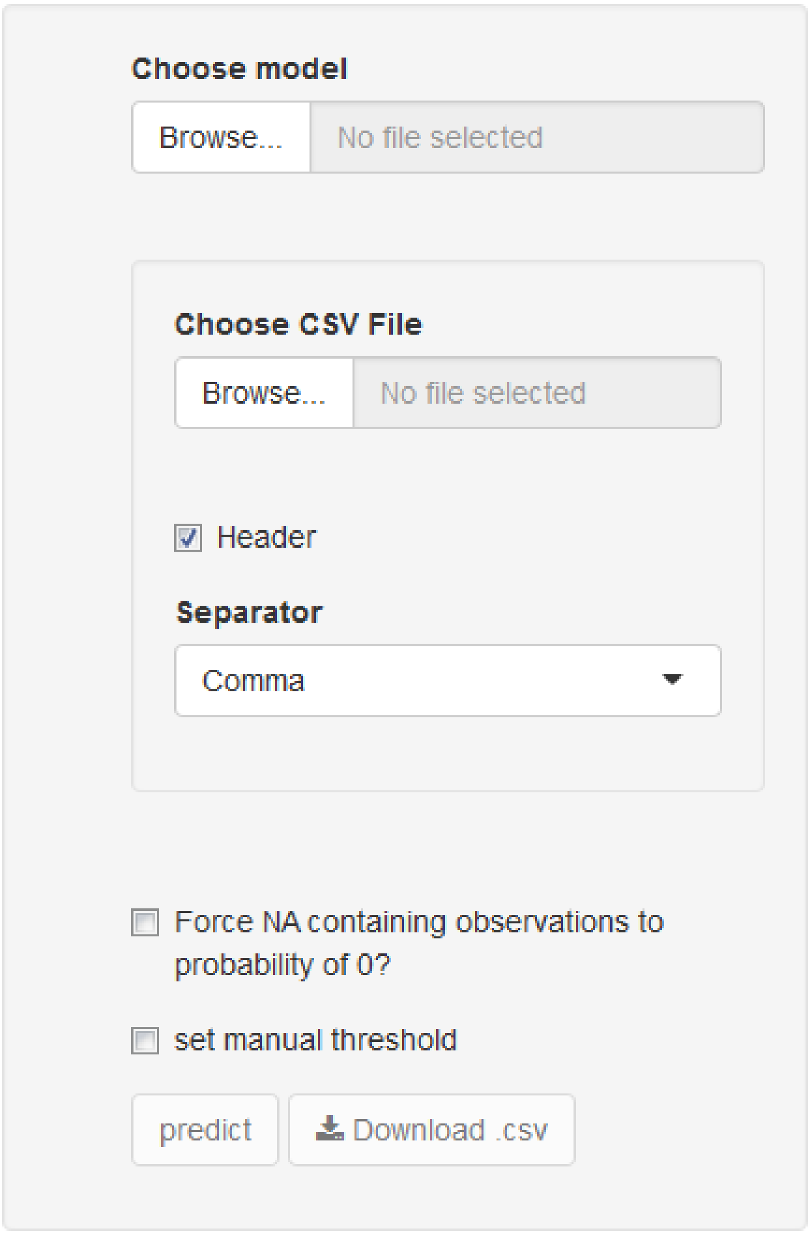


Figure 6: Sub menu for the tab Validate model

3.2.6 Predict

The Predict tab enables the prediction for new data with the uploaded model. The well panel enables the upload of the new dataset similar to the functionality in Import Data. Furthermore, there are a two advanced settings options for the prediction.



Choose model

Browse... No file selected

Choose CSV File

Browse... No file selected

☑ Header

Separator

Comma

☐ Force NA containing observations to probability of 0?

☐ set manual threshold

Figure 7: Sub menu for the tab Predict

3.2.7 Others

The functionality available here is the upload of a plot function. This function has to start with "plot_" and need to be able to generate a plot from one line of the uploaded dataset. This plot function is used everywhere in the application to visualize data. Furthermore, also all model parameters and if applied generated feature calculation functions with their exact calls can be found here.

4 Using the Software

Using the Software section provides a detailed description of software functions.

4.1 Import data

This tab represents the central data input for a new model. Under "Choose CSV File" *.csv* files can be uploaded. In case more than one file is uploaded the layout of all uploaded files has to be the same. If this criteria is fulfilled all *.csv's* are joined together to one dataset. The *Header* checkbox specifies whether the uploaded files contain a header that specifies the content of the columns or not. The *Separator* drop down menu specifies the separator that is used in the uploaded *.csv* files. Whereas *Quote* specifies the quotes that are used to quoting characters in the uploaded files. If the upload of the data set has worked a data table appears in the main panel displaying all observations with their corresponding features.

4.2 Calculate features

The *Calculate features* tab is a convenient interface to calculate new features from the uploaded dataset. The rational behind the calculation of new features is inter alia to connect single features (*e.g.* relative response values through a fitted model) with each other since algorithms like *randomForest* treat only one feature at a time and do not know anything about the connection of those features. In principle this does not need to be done in the CurveCassification environment but the advantage here is that the used functions were linked to the model, if its performed in this environment. This enables the model for future predictions to perform all prepossessing steps automatically, which has to be done otherwise manually. The user has the ability to upload specific feature generation functions that can be implemented. Also a variety of default feature generation functions are available that can be used directly or with slightly adaptions to the data structure.

1. fgf_auc: Calculation of the area under the relative response values.
2. fgf_polynomFit: Fitting of a polynomial of a specified degree (default = 1). Parameters as well as the R^2 are returned.
3. fgf_slopeCounts: Calculates the mean slopes for the specified data points.
4. fgf_diffVar: Calculates the difference between a variance value for a observation with the respective response values.
5. fgf_sumPositiveSlopes: Counts the positive slopes between the specified values.
6. fgf_meanSlopes: Calculates the mean slopes between the specified values.

7. fgf_normalize: Normalizes the selected values to the 0 nM / DMSO concentration intensity.

8. fgf_nonlinearfit: Fitting of a log-logistic model with the drm function from the package *drc*. (Ritz, C., Baty, F., Streibig, J. C., Gerhard, D. (2015) Dose-Response Analysis Using R PLOS ONE, 10(12), e0146021)

A detailed description of every function could be found in section 4.2.7.

4.2.1 Choose R script: User defined feature generation functions

As mentioned before besides of those predefined feature calculation functions under the item *Choose R script* the user can upload a R-script that contains functions. The new function names need to start with fgf_. Otherwise those functions were not available in the "Curve-Classification" environment. There are three general requirements which have to be fulfilled from such functions. The first is that the first parameter of the function have to represent the dataset. Furthermore, one parameter of the function has to be "pattern". The underlying idea here is that "pattern" represents a regular expression that is used to select columns of the dataset on which the function should be applied. However, "pattern" has to be only present in the function definition as a parameter and does not have to be used in the function if the syntax of the function does not suit the user defined function. The last requirement is that the function has to return the whole dataset including the new calculated features. As an example the code for the fgf_normalize function is displayed below.

```r
fgf_normalize <- function(data , pattern){

  colnames(data) <- gsub("\\.DMSO", ".0nM", colnames(data))

  idxDMSO <- grep(paste0(pattern , "0nM") , names(data))

  idxPattern <- grep(pattern  , names(data))

  normalizedData <- apply(data , 1 , function(x , idxDMSO , idxPattern){

    x[idxPattern] <- as.numeric(x[idxPattern])/as.numeric(x[idxDMSO])

    return(x[idxPattern])

  } , idxDMSO = idxDMSO , idxPattern = idxPattern)

  data[ , idxPattern] <- as.numeric(t(normalizedData))

  return(data)

}
```

4.2.2 Feature calculation function

Here all feature generation functions are available to be selected. This includes the predefined functions as well as the potentially uploaded feature calculation functions. Here one or multiple functions can be selected, but one function can be only selected once. The argument "pattern" is specified with the next input field "Feature generation function patterns".

4.2.3 Feature generation function patterns

This input field takes text stings that are passed to the selected functions as pattern argument. There are two different ways to specify patterns. If only one pattern should be applied to all previous selected functions, only this pattern has to be written into this field. The second option is that for every selected function a specific pattern should be applied. To do this the number of patterns and functions has to be the same and the checkbox *Feature generation function patterns* has to be checked. The different patterns than have to be separated with a semicolon without any spaces in-between and the number of patterns has to be the same as the number of functions. The order of the functions also represents the order in which they were executed.

4.2.4 Calculate features

Once all settings were made clicking onto the button *Calculate features* execute the selected functions with their patterns and the uploaded dataset. If an error occurs it would be shown in the field "Messages" in the main panel. If all features could be calculated without any error a table appears in the main panel in which every row represents the summary of one column of the initial dataset. After the features are calculated the feature calculation procedure can be repeated for other functions or the same functions with other patterns. All those functions were memorized, and therefore, the result is the same whether all features are calculated at once or they were calculated separately.

4.2.5 Plot options

An additional functionality that is available in this tab is the visualization of the features. By clicking on a row in the table that appears after the feature calculation was executed, a bean plot with two distributions appears under this table. One of those distributions is labeled as positive and one as negative. The decisions which variable is used to split could be made by selecting the variable in the *Select target* drop down menu. In the subsequent drop down menu *Select positive class* the value of the target column that should be treated as positive could be selected. If the target column has more than two values the selected positive is treated as positive and all others are assigned to the negative class. This functionality enables the user to estimate the potential of a feature to separate the classes.

4.2.6 Use data with new features

By clicking onto the button *Use data with new features* the initial uploaded dataset is replaced by the dataset that contain the additional calculated features. Subsequently, the table with the summary of the columns disappears as well as the plots.

4.2.7 Details about the default feature generation functions

The functions expects concentrations that are expressed in nM, otherwise it wont work and the concentration of 0.nM needs to be named instead of 0.nM as DMSO.

fgf_auc This function calculates the area under the responses of the response values. The values between two responses are linearly interpolated using the function *auc* from the *MESS* package.

If the specified pattern represents more than two concentrations the concentration of DMSO is replaced with 1/10 form the minimal value. This is done due to the fact that the area should be calculated with logarithmized concentrations. If the specified pattern represents less than two concentrations the concentration of 0.nM remains unchanged and the AUC is calculated for the unlogarithmized concentrations.

If more than three concentrations are available the values are smoothed by a moving average of length three before calculating the AUC.

The new feature that is calculated is named with the name of the applied pattern combined with *AUC*.

fgf_polynomFit This function performs a polynomial fit onto the response values specified with pattern. Per default a polynomial of degree one is used. Besides of this the concentration of 0.nM is only used if only two concentrations are specified with the pattern and additionally the degree is one. Else the polynomial is fitted to the logarithmized concentrations (excluding 0.nM) and their respective responses.

The function returns all parameters of the fit, except of the intersect, as well as the R^2. It could be shown that for the datasets used in the study ("Manuskript") the R^2 correlated with the one obtained form the fit of a log-logistic model. Therefore, this function can be a fast alternative compared to the fitting of a log-logistic model.

fgf_slopeCounts This function is intended to calculate the mean slope for the applied feature (usually used for counts like MS/MS counts). However, it is assumed that the distances of the x-values are equal. Therefore, this feature should be only applied to count the directions of slopes.

The value that is returned form this function is named *Slope.* and the applied pattern.

fgf_diffVar This function calculates the difference between a variance value and the respective response values. The intention to integrate this function was the problem of missing values in the variance caused by cases where a protein is only measured once.

As, missing values are an issue for almost all classification algorithms this feature can not be used directly. A way to generate a value, which captures the information of the variance and addresses the problem of missing values is a transformation. The initial variance is subtracted by one and the minimum of the relative response of the specified concentrations is added onto this value. This results in a negative value if the error bar of variance is not overlapping with the minimum of relative response and a positive value if there is a overlap. For all missing values zero was assigned for the transformed variance, which represents an intermediate level of quality.

fgf_sumPositiveSlopes This function counts the slopes of the lines between the responses of the specified concentrations that are greater than zero. It returns a single value that is named with the applied pattern in combination with *SumPositiveSlopes*.

fgf_meanSlopes This function calculates the mean slopes between every specified concentration and all other concentrations based on the logarithmized concentrations. This means that the number of returned values is the number of applied concentrations - 1. The basic idea is to reduce the influence the outlyars at particular concentrations.

fgf_normalize This function normalizes response of the specified values onto the DMSO / 0.nM concentration. Therefore, all values specified via pattern are divided by the concentration of 0.nM to generate relative values.

fgf_nonlinearfit This function performs a least square fit of a log-logistic model onto the specified pattern. Here the *drm* function from the *drc* package is used. The used function is a LL.4 function which has the four parameters: Top, Bottom, Slope and Inflection. Those four parameter and the R^2 were returned. However, this function is slow due to the fitting which is slow.

4.3 Generate new model

The aim of the tab is to generate a model from the uploaded and maybe processed dataset.

4.3.1 Exclude features

Since a uploaded dataset does not contain necessarily only features that are relevant for target classification the *Exclude features* selector gives the ability to exclude irrelevant features. Features that are not relevant for the target prediction occur often if the raw output from omics analysis is used (*e.g.* Maxquant ProteinGroups.txt). Therefore, it is important to have an idea which features were relevant for the given classification task. As those datasets were usually initially judged manually, the criteria used there are a reasonable choice for a first model.

4.3.2 Select target

In this selector one feature/column has to be selected as a target column, which indicates the column that is to be predicted by the model. The selected feature here is excluded anyway for model training so it does not matter if the target column is selected under *Exclude features* or not.

4.3.3 Select positive class

In this selector all values that occur for the selected target feature (made under *Select target*) are selectable. Only the value that is selected here is treated as positive (*TRUE*) whereas all other values, no matter how many there are, were treated as negative(*FALSE*). This is due to the implemented infrastructure which allows only a binary classification. It is common in machine learning approaches to assign the smaller class as positive, since some performance measures depend on the definition of the positive class.

4.3.4 Advanced settings

There are a few settings that can be used to further optimize a model. However, the default values work also well and are sufficient for a first fitting of a model.

Ratio to split data into train and test With this selector the proportion of train and test data could be adjusted. The default of 0.8 defines that 80% of the data were assigned as training data whereas the remaining 20 % were used as test data. This value is common in machine learning approaches and should be only changed in justified individual cases.

Select a undersampling rate With this selector the rate could be defined with whom the bigger class is undersampled. The default is calculated according to the ratio of positives and negative observations for the target variable. The default value represents a value of 1 : 3 (positive : negative observations) if possible. There can be some cases where the adjustment of the undersampling rate focuses the model more onto a specific class; but in general the default values work well.

Tune threshold? By selecting the check box another selector appears that is named *Tpr tune value*. The value selected here represents the true positive rate which should be reached by the model. Therefore, the threshold that is necessary to ensure this true positive rate is estimated from the train dataset and stored in the model. This threshold is accessible in the performance plots in the main panel as black

line as well as under *Model parameters*. However, the threshold could be manually adjusted under *Predict* for every prediction made. As a true positive rate of one could cause a very high false positive rate the upper bound for the *Tpr tune value* is limited to 0.995.
If the check box is unchecked no threshold is estimated and the default of 0.5 is used unless the user specifies another threshold manually for every prediction under *Predict*. A consequence is that all performance measures in the main panel are calculated based on a threshold of 0.5, if the threshold is not optimized. Otherwise the threshold is adjusted to the estimated value.

4.3.5 train

By clicking train a model is trained with the selected parameters above. The time that is needed for the training depends on the dataset size as well as the undersampling rate used. In general the training take 5 to 10 minutes. After the training of a model was successful a message in the *Messages* field in the main panel is shown that says in green "Model trained successful", if an error occurs during the training a specific error message is shown in red in the same panel.

4.3.6 Download full model

If a model was trained successfully this model can be downloaded here. This model can be used subsequently to predict classes for new data. Furthermore, this model contains all observations used for the training of the model, and therefore, operations like nearest neighbors can be performed.

4.3.7 Download model for pipeline

This button is in principle the same like *Download full model*. The only difference is that the data used for the training of the model is not kept in order to ensure a fast loading if this model should be used within a pipeline. Thus, this models have only a limited functionality within the application. With such a model no nearest neighbors can be found. Furthermore, the optimization of a model is only possible with a limited number of features. Good advise is here to download this model twice once for pipeline and once a full model.

4.3.8 Main panel

Messages The panel *Messages* shows all messages from the underlying functions and algorithms that are relevant for the user. This contains messages that confirm the successful performance as well as error messages that occur.

Performance Under the tab performance the performance of the current model is available.

Confusion matrix Here the confusion matrix from the data that is assigned to the test set is shown. The number of false positives, false negatives, true positives and true negatives are depicted.

Performance measures Here the true positive rate, false positive rate, accuracy, area under the ROC curve and precision are shown.

Performance vs. Threshold Here a plot of the true positive rate, false positive rate, accuracy and precision is shown for a varied threshold. The user can estimate how robust a model works. The line represents the used threshold.

Probability distribution Another plot depicts the probability distribution predicted by the model for positive and negative observations, respectively. The black line also represents the applied threshold.

Data Here a data table with the used data set is shown as well as the prediction from the model. All observations are selectable by clicking onto the corresponding row. In case a plot function is available (could be uploaded under *Others/Upload plot function*) those observations are visualized under the data table. This enables the user to check the false positives and false negatives, since the data table is subsetable. Furthermore, the user can get a feeling of the decisions made by the model.

Nearest Neighbors This is almost the same as *Data* content wise. However, a click onto a row does not only visualize the selected observation. The nearest neighbors within the data set are searched that are TRUE and FALSE, respectively; and subsequently visualized. The nearest neighbors are searched based on all variables (centered and scaled) that are included into the model using a euclidean distance matrix. Therefore, a dataset can be checked for consistency because it was observed that the decisions which are made during manual classification are often inconsistent. This functionality enables the user to critically rejudge such observations and maybe reevaluate some in the training dataset.

4.4 Optimize existing model

The aim of this tab is to optimize an existing model. This could be done in two general ways.

The first way is intended for an early stage of the generation of a model, where the initial dataset was partially reevaluated. For example the predicted false positives and negatives are reevaluated using the visualization of the application. Here the usage of *Optimize existing model* is not mandatory, since the same procedure as for the initial model can be repeated. However, using this tab saves time as the calculation of feature generation functions and all other settings are taken from the old model. To do this the check box *Keep old training and test data* needs to be unchecked.

The second way is the integration of additionally manual judged data. Here a dataset can be used which was not used for the model that is to be optimized. However, this dataset could be predicted with the model with a subsequent manual evaluation of this prediction. If the structure of the dataset is then the same as from the initially used data set this can be used to focus the model a bit more onto a certain feature for example. This dataset is then added to the initially used dataset that is stored in this model. The new dataset is split with a ratio of 5:1 into training and test data. To do this the check box *Keep old training and test data* needs to be checked.

While the first option works for all models generated with this app the second is only possible if the model is a full model (contains the dataset used for training and testing).

4.4.1 Choose model

Here a model can be uploaded that is generated by this application. Such a model is file of type*.RData*.

4.4.2 Choose reannotated data

Here a new dataset could be uploaded, depended on the option that should be used to optimize the existing model. However, the other functionality of this well panel is similar to the one introduced in the section *Import Data*.

4.4.3 Tune threshold?

By selecting the check box another selector appears that is named *Tpr tune value*. The value selected here represents the true positive rate which should be reached by the model. Therefore, the threshold that is necessary to ensure this true positive rate is estimated from the train dataset and stored in the model. This threshold is accessible in the performance plots in the main panel as black line as well as under *Model parameters*. However, the threshold could be manually adjusted under *Predict* for every prediction made. As a true positive rate of one could cause a very high false positive rate the upper bound for the *Tpr tune value* is limited to 0.995.
If the check box is unchecked no threshold is estimated and the default of 0.5 is used unless the user specifies another threshold manually for every prediction under *Predict*. A consequence is that all performance measures in the main panel are calculated based on a threshold of 0.5, if the threshold is not optimized. Otherwise the threshold is adjusted to the estimated value.

4.4.4 Keep old training and test data

This check box specifies whether the actual training and test data of the model should be kept or not. This check box specifies which kind of optimization should be performed for the current model. The two types are explained in the introduction of this tab.

4.4.5 optimize

By clicking *optimize* the model is retrained with the new data basis while keeping all other settings from the model that is to be optimized. After a successful training new buttons appear. Two buttons are to download the optimized model (similar to *Generate new model*), whereas the third button *Use new model* regulates the exchange of the current and the new model.

4.4.6 Use new model

By clicking onto this button the previous model is replaced with the retrained model. Therefore, this model is used for prediction for future models.

4.4.7 Main panel

The general actions that can be performed in the main panel are similar to the one that are available under *Generate new model*. The only difference is the performance tab. Here are the performances of the new and the old model are faced. The underlying test dataset is the test data from the new model. This means tow model performances can be compared on the same test dataset.

4.5 Validate Model

In this tab all false positives and false negatives can be visualized if a plot function is available. This is a more convenient way if all false positives and false negatives should be visualized, compared to manually selecting every false positive and false negative observation under *Generate new model/Data* or *Optimize existing model/Data*, respectively. The radio buttons under *Select a model* specify for which model the false predicted should be visualized; the one available under *Generate new model* or *Optimize existing model*.

4.5.1 only for test data

If this box is checked (default) the false predicted observations are only visualized for the test dataset, whereas otherwise the false predicted observations are visualized for all observations.

4.5.2 Generate plots

By clicking onto this button the plots are generated for the false observations for the selected model and the selected dataset.

4.6 Predict

In this tab a generated model can be used to predict the classes for a new dataset. Furthermore, those observations of the new dataset can be visualized. Also the nearest neighbors of a new observation, in the data set the model is based on, can be found and visualized.

4.6.1 Choose model

Here a model can be uploaded that is generated by this application. Such a model is file of type*.RData*.

4.6.2 Choose CSV File

Here a new dataset could be uploaded. The other functionality of this well panel is similar to the one introduced in the section *Import Data*.

4.6.3 Force NA containing observations to probability of 0?

It could be that the dataset that is to be predicted contain one or more missing values for variables that were used by the model. Since no imputation method is applied to the dataset the model is not able to predict the class for such observations. By selecting *Force NA containing observations to probability of 0?* those observations were predicted as negative with a probability of zero. However, by choosing this the information about which observations does contain missing values is lost. If this check box is unchecked all observations that contain a missing value in a variable required for the prediction is predicted as NA. This conserves the information about which observations does contain missing values.

4.6.4 Set manual threshold

By selecting this check box another selector appears in which the threshold that is to be applied can be specified. Values between zero and one are possible and the default is 0.5. If this box is unchecked the threshold stored in the model is applied. In case the threshold is estimated during model generation or optimization this threshold is applied and otherwise the default threshold of 0.5.

4.6.5 Predict

This button can be clicked if a *.csv* file is uploaded and if a model is available. By clicking onto this button the model is predicted using the uploaded model. After a successful prediction the *.csv* file appears as data table with two additional columns probability and prediction.

4.6.6 Download .csv

By clicking onto this button the uploaded *.csv* file with the additional columns probability and prediction is downloaded.

4.6.7 Main panel

prediction After a successful prediction a data table appears in this tab that contains the uploaded *.csv* file with the additional columns probability and prediction. If a plot function is available, a plot appears under the data table for the selected rows (by clicking onto them).

Nearest Neighbors This tab depicts a similar functionality as the Nearest Neighbors available under *Generate new model/Nearest Neighbors* or *Optimize existing model/Nearest Neighbors* with the difference that here the nearest neighbors are searched within the dataset that is used to generate the model. This means that this functionality is not available if a model is used that is for pipeline integration and does not contain the dataset on which it is based.

4.7 Others

4.7.1 Upload plot function

In this tab a plot function can be uploaded in order to visualize observations. The script that contains the plot function can contain a variety of helper functions and variables. However, the function that generates the final plot needs to be named starting with *plot_*. This function has to have the ability to generate a plot from a single observation(one row of the dataset). Besides of this the only limitation is that the first argument has to be data. As mentioned before the plot function obtains a single row from the dataset. The script can be uploaded under *Select a plot script* and has to be a *.R* file.

If a plotfunction is uploaded all elements that are defined in the script are listed under *Objects sourced from the script*. Under condition that a dataset is uploaded also a Testplot appears in order to get a first idea of the uploaded function. Under *Plot function* all objects that were defined in the script are printed, in order to get an idea of the used plot function.

There is a default plot function called *Default_Plotfunciton.R* which generates a barplot of all feature names that end with an capital "M", since concentrations end usually with an "M". However, this function can be easily changed to plot any subset of features needed. The pattern argument specifies which features are to be plotted.

4.7.2 Model parameters

In this tab all the model parameters are accessible under *Hyperparameters* all parameters that need to be specified in order to generate a model using the mlr package are visualized. Furthermore, *Used features* depicts all variables that are used from the dataset to generate the model.

Under *Feature generation function calls* all functions that are used to generate new features with their exact calls and the whole source code for those functions is accessible.