Documentation - Modeling Toolkit

# Background and Introduction

The Modeling Toolkit is intended to serve as a holistic machine learning solution that integrates both visual analysis and modeling functionalities into one user-friendly app. The provided functionalities should enable a structured procedure for the modelling of facts. Background and main claim of the toolkit is above all the democratisation of machine learning methods by providing an intuitive user interface and should not be confused with   
State-of-the-art procedures in the field of Supervised / Deep Learning compete.

# preliminary considerations

Before using the application, the intention and first hypotheses of the modeling should first be considered and formulated. The following questions can be used to support your thoughts:

* *What is my problem or context that I want to transform into a model?*
* *Which indicators / measurement variables do I generally consider for this? Do I use the variables to measure what adequately reflects my actual situation?*
* *What is the purpose of the model? Does it primarily serve to explain / statistically validate the factual context or to predict new data points?*
* *To what extent are the selected indicators measurable? Are they sufficiently detailed, complete and often surveyable against the application context?*
* *Which resources are connected with the modelling or the construction of the necessary data infrastructure?*

These questions will be used to outline a rough business case for modeling. There should be a logical fact for a modelling that promises entrepreneurial added value and can be implemented with realistic resources. These preliminary considerations are decisive to ensure practical modelling and its implementation within the associated organisation.

# APPLICANCE OF THE MODELING TOOLKIT

Basically, the modeling process can be divided into two phases:

* data exploration phase
* data modeling phase

## Explorative data analysis

During the exploration phase, potentially suitable explanatory variables for modelling the target variable must be identified and interfering variables eliminated. Furthermore, distributions are to be identified which violate fundamental modeling assumptions and have to be transformed in downstream steps (e.g. variables with "oblique" distribution). Depending on the type of modeling, the individual steps should be weighted differently. Although this function can be basically automated (e.g. via [feature elimination](https://medium.com/@sagar.rawale3/feature-selection-methods-in-machine-learning-eaeef12019cc)), it is recommended to get an "overview" of the data or the distributions and correlations of the underlying variables. The *plotting box* on the left side of the application is used for this purpose. Two fundamental exploration steps can be carried out within the box:

* Descriptive frequency & context analysis
* correlation analysis

Using descriptive frequency analysis, a variety of graph types (density plots, bar plots, box plots, heat maps, scatter plots) can be created that can be used to analyze the data structure. Similar to a drag & drop system, the variables or their frequencies can be arranged on an X & Y variable. A third grouping variable can also be used to identify specific group differences within the distribution (NOTE: The grouping variable must have a categorical scale level). The following table provides an overview of the available visualization options:

|  |  |  |  |
| --- | --- | --- | --- |
| *Y* | *X* | *group* | *plot* |
| metric | - | Not available | **Density plot** |
| metric | - | Available | **Grouped Density Plot** |
| metric | metric | Not available | **Scatter Plot** |
| metric | metric | Available | **Grouped Scatter Plot** |
| metric | categorical | Not available | **Box Plot** |
| metric | categorical | Available | **Grouped Box Plot** |
| categorical | - | Not available | **Bar Plot** |
| categorical | - | Available | **Grouped Bar Plot** |
| categorical | categorical | Not available | **heat map** |

Within the second tab of the plotting box, the relationships of all variables with metric scale levels are visualized (categorical variables are not included, since no correlation analysis is possible for them). The aim of the correlation analysis is to identify variables with a strong (positive as well as negative) correlation to the target variable and to exclude multicollinear variable pairs (correlation > 0.75) for further analysis.

## data cleansing

In summary, the exploration phase is intended to provide a holistic understanding of the data in order to make a preliminary selection of suitable modelling variables and estimate necessary pre-processing measures. The next step is to define the necessary transformation steps resulting from the previous data exploration - for example, the treatment of skewness. The following procedures have been integrated within the preprocessing tab for the most frequent transformation requirements:

|  |  |
| --- | --- |
| *method* | *description* |
| **standardisation** | Scaled mean value of 0 and standard deviation of 1 |
| **normalisation** | Scales values between 0 and 1 |
| **log transformation** | Applies the natural logarithm to values |
| **Yeo-Johnson- & BoxCox transformation** | Method for approximating a normal distribution and stabilization of variance |

Since only variables with a metric scale level are permissible for modeling, it is necessary to bring relevant categorical variables into a suitable format. These are converted into binary variables (0-1) using encoding methods. The following methods have been integrated into the toolkit:

|  |  |
| --- | --- |
| *method* | *description* |
| **one-hot encoding** | Formation of **n** binary variables with n categorical values |
| **dummy encoding** | Formation of **n-1** binary variables for n categorical expressions (last expression is defined by the non-occurrence of the others) |

Furthermore, the app provides suitable functions for the professional handling of outliers and missing values. The user can choose between the following treatment options, which are characterized by various advantages and disadvantages depending on the application context:

|  |  |  |  |
| --- | --- | --- | --- |
| **<** | *scale level* | *function* | *description* |
| **Missing values:** | Metric | ***delete*** | Delete observation with missing values |
|  | Metric | ***median*** | Replace missing values with the median of the variable |
|  | Metric | ***mean*** | Replace missing values with mean value of variabl |
|  | Discreet | ***keep*** | Delete observation with missing values |
|  | Discreet | ***fashion*** | Replace missing values with variable mode |
|  | Discreet | ***random*** | Replace missing values with random values |
|  |  |  |  |
| **Outlier:** | Metric | ***keep*** | No treatment of outliers |
|  | Metric | ***median*** | Replace Outliers with Median |
|  | Metric | ***mean*** | Replace outliers with mean value |

Last but not least, the application offers various filter functions to remove variables with properties,   
which may result in an impairment - if not prevention - of the training process:

|  |  |
| --- | --- |
| *method* | *description* |
| **multicollinearity** | Strong correlation between predictors |
| **linear combination** | Predictors represents the linear combination of another predictor. |
| **Variance homogeneity** | variable has a variance of (nearly) 0 |
|  |  |

## Modeling

Once the transformation measures preceding the modelling have been defined, it is necessary to select the modelling algorithm (or algorithms) to be used. In principle, the decision as to how many model types should be trained at the same time is based on a trade-off between performance and computing power. The data size is usually the decisive criterion for weighing up this trade-off. With increasing data volume, the accuracy and generalizability of the model usually increases, but this also increases the time and computing effort per training period. As a rule of thumb, it is recommended to make a selection of 2-3 algorithms based on the table below, which could potentially be successful for the specific problem. For extremely large amounts of data (data sets > 1 TB) it is recommended to include only one algorithm per training period. The following algorithms are available for the current status:

|  |  |  |
| --- | --- | --- |
| *label* | *algorithm* | *problem type* |
| **lm** | Linear Regression Algorithm | regression |
| **Ida** | Linear Discriminant Analysis | classification |
| **ada** | Ada Boosting Algorithm | Regression, Classification |
| **rf** | Random Forest Algorithm | Regression, Classification |

A more detailed documentation of the algorithms including their advantages and disadvantages of the different models can be found under [the following link](https://topepo.github.io/caret/available-models.html). So-called ensemble algorithms (such as rf, ada), which are characterized by extremely high flexibility in the area of application and simultaneously high accuracy, have proven themselves as "all-rounders". After the final selection of the modeling algorithms, the procedure within the training process must be specified more precisely. A decisive step here is the selection of the indicator according to which the model is to be optimized. Different indicators are available to the user for each problem:

|  |  |  |
| --- | --- | --- |
|  | *problem type* | *explanation* |
| [**accuracy**](https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce) | classification | # correct predictions / # predictions |
| [**Cohen's (unweighted) Kappa**](https://thedatascientist.com/performance-measures-cohens-kappa-statistic/) | classification | Statistics for the Interrate-Reliability Check |
| [**R²**](https://towardsdatascience.com/statistics-for-machine-learning-r-squared-explained-425ddfebf667) | regression | Declared variance / Total variance |
| [**RMSE (Root mean squared error)**](https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234) | regression | Standard deviation of residuals |

Another relevant aspect is the "tuning" of hyperparameters - i.e. those parameters that the algorithm cannot optimize itself and therefore have to be selected by humans. In practice, it has been proven that this selection process follows a classic trial & error approach, which is automated within the training process. Common procedures were integrated into the app:

|  |  |
| --- | --- |
| *method* | *description* |
| **Random Hyperparameter Search** | Random selection of parameter values from a defined value range |
| **Grid Hyperparameter Search** | Determination of specific parameter values |

If the Random Approach of the Hyperparameter Tuning is selected, the Toolkit sidebar will display sliders for the selection of the parameter ranges to be included. Due to the complexity of these parameters, this should only be considered in specific cases. The third decision component within the training specification is the definition of the Train & Test Set size. This method is based on the common principle within data science practice of dividing the existing data set into two distinctive components. The so-called Train-Set serves to optimize the model. The test set, on the other hand, is a kind of insurance for the practical suitability of the "trained model". It has not yet been "seen" by the algorithm and therefore reflects good comparability with new data, which will later also be used for classification in practice. The usual ratio of "80% training - 20% testing" is preset as standard within the application and can be adapted at will. Another obvious determinant is the selection of the validation process used in hyperparameter tuning. The current status can be selected from the following validation procedures:

|  |  |
| --- | --- |
| *method* | *description* |
| **boat** | bootstrapping |
| **cv** | K-Fold Cross Validation |
| **repeatedcv** | Repeated K-Fold Cross Validation |
| **LOOCV** | Leave-One-Out Cross Validation |
| **LGOCV** | Leave Group Out Cross Validation |

All of the above procedures require the setting of resampling iterations - i.e. the number of times the result is to be validated. Similar to the number of model algorithms included, the calculation time increases with each validation loop. The selection should therefore be carefully considered and weighed in relation to the data size or accuracy requirements of the customer. Individual validation procedures also offer granular setting options. A detailed explanation of the procedures and their parameters can be found at the [following link.](https://towardsdatascience.com/validating-your-machine-learning-model-25b4c8643fb7)

After cleaning up the data set and selecting the training parameters and algorithms to be integrated, the modelling can be started. Since even small errors can prolong the training duration unplanned, it is advisable to carry out a final sanity check in advance to check the correctness of the parameters. During the training process, a status bar appears at the bottom right, which serves as a rough indication of the progress of the training process. Since unfortunately no more concrete time specification can be represented, it is recommended to carry out iterative modelling periods with increasing complexity (e.g. by more validation steps or inclusion of several algorithms) in order to get a rough feeling for the "expenditure of time". This ensures that no more time and computing power is invested than is absolutely necessary. If a training process takes an unexpectedly long time, it can usually be stopped by closing the app. In rare cases it can happen that the application is not closed and remains active in the background. It is therefore advisable to check the actual closing of the program in the Task Manager.

## model evaluation

After the training of the selected model options is completed, you can proceed to the performance analysis. The "Performance Box" in the right part of the dashboard is decisive here.   
The first tab serves as a cross-model overview in which all indicators relevant to the problem are displayed for each algorithm. The "Model Metrics" tab lists more detailed performance information for the individual models, which is explained in more detail below:

|  |  |  |
| --- | --- | --- |
| *method* | *problem type* | *explanation* |
| [**Confusion Matrix**](https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234) | classification | See link |
| [**Group-specific Metric Plot**](https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce) | classification | See link |
| **Density Comparison Plot** | regression | Forecasted VS Actual distribution |
| [**Residual plot**](https://towardsdatascience.com/how-do-you-check-the-quality-of-your-regression-model-in-python-fa61759ff685) | regression | See link |

The other two tabs contain indicators for model interpretability. A more detailed explanation of [global](https://towardsdatascience.com/hands-on-global-model-interpretation-3bb4264732b5) and [local](https://gilberttanner.com/blog/local-model-interpretation-an-introduction) interpretation methods of models can be found under the attached links. A major difference between the two methods is that global methods attribute a general significance to the variables used for decision making, whereas local methods focus on the decision-making mechanisms and the influence of variables on individual characteristics. Within the "Variable Importance" tab, the general explanatory contribution of the variable is visualized, which can be considered as a global interpretation indicator. Within the tab "SHAPLEY Values" the decision mechanisms of the model for the first 100 cases of the test set can be visualized. These can be derived from the SHAPLEY values presented for the case under review and serve as a local interpretation indicator. Within regression models, these can be interpreted as a (positive/negative) influence on the value of the target variable. For classification models, especially those with a multinomial nature (more than 2 categories to be predicted), a different interpretation of the SHAPLEY values applies. These can be interpreted approximately as a (positive/negative) influence on the probability of the respective expression and must therefore be assessed for each class of a categorical target variable.

With the exception of the overview tab, the graphics can be viewed for all algorithms included. The selection of the corresponding model is done by the corresponding selection field in the header of the application. In addition to the tab, the performance indicators of the selected model are summarized in the four upper KPI boxes.

## Model selection and export

The visualized graphics, especially within the "Performance" box, are intended to provide a holistic overview of the quality of the individual model options. The first two tabs and the KPI boxes provide a detailed insight into the forecasting accuracy of the options, whereas the latter tabs provide a deeper insight into the forecasting mechanisms of the model. Together they should form the information foundation for carefully weighing up model interpretability and accuracy within the context of the respective project and selecting a preferred model.

After successful decision making, the Modeling Toolkit offers a wide range of options for the practice-oriented further use of the model. On the one hand it offers the possibility to export the model as well as previous pre-processing steps by clicking the "Download Model" button in a . zip file. The data is transformed into the ".rds" format, which can be read by all R programs. Using the corresponding script "XXX.R" new values can be classified without problems. The file path to the model and pre-processing steps as well as the new data set within the script have to be adjusted beforehand. In addition, it is possible to embed the exported model in an interactive "prediction tool", which offers an intuitive user interface similar to the Modeling Toolkit itself. The prediction tool allows you to open a data set (via database or file upload) and automatically classify the data using the embedded model. An analysis of the variable distributions as well as the prognosis mechanisms (via SHAPLEY values) can be performed under the corresponding tabs. The export of the predicted data set within the Prediction Tool takes place via CSV export.

Furthermore an automated model report with interactive graphics can be downloaded by clicking the "Download Report" button. The user only has to specify the project name, background and reason for the model decision in the pop-up menu.

# Transferring the Model to the Prediction Toolkit

The transfer to the Prediction Toolkit follows a standardized process, which is listed in more detail below:

1. Copy the template folder and rename it according to the project context
2. Change the concrete name of the toolkit as well as the corresponding color palette (consisting of main color and secondary colors, in HEX format) within the "ui.R" file in the corresponding variables. Since a large part of the user interface is designed in the main color, a color with as strong an association as possible - such as the corporate color - should be used. The secondary colors are used within the graphs - especially in the visualization of group differences - and should therefore be sufficiently distinguishable. The number of secondary colors required is usually determined by the categorical variable with the most characteristics within the data set used for modeling. As a rough rule of thumb, at least 4 distinguishable secondary colours should be used.
3. If a logo branding is desired, add a corresponding "logos.png" and "logos.ico" file to  
    the subfolder "WWW" of the template folder. The PNG is used top right within the toolkit and as the ICO as the general logo of the application. On the [following website](https://www.zamzar.com/convert/png-to-ico/) PNGs can easily be converted into the appropriate icon format.
4. Insert the corresponding ZIP file of the exported model Toolkit based on the models into the Template folder.
5. Now the file can be converted into a Windows installation file. For this the R.file "prediction\_tool\_deployment.R" must be opened. First select lines 1-3 and execute (CTRL + ENTER). The necessary packages are installed and activated. The X-X lines contain important parameters of the installation file, which can be adapted according to the project context (e.g. renaming the application or changing administration authorizations):

|  |  |  |
| --- | --- | --- |
| **app\_name** | STRING | Name of the Application |
| **app\_dir** | STRING | Directory of the App Folder (Update Accordingly) |
| **include\_R** | BOOLEAN | Indicate if Installer should contain a version of R (Required for all PCs without R installed) |
| **privilege** | STRING | Setup Installation Requirements - “high”: Administration Status required - “low”: Everyone can install Application |
| **default\_dir** | STRING | Default Installation Directory for Application (see XXX for further details) |
| **app\_icon, setup\_icon** | STRING | Name of Application / Setup Icon  (suggested to keep naming and overwrite “app\_icon.ico” with a custom icon the main folder) |
| **pkgs** | VECTOR | List of required packages to launch the App (If no major changes to the app are applied, keep the list) |
| **electron things** |  | VS FIREXFOX OR CHROME |

1. After the parameters for the application have been adjusted and the "INNO" program has been installed, the "runApp(...)" command can be used to ensure that the application functions as expected and that no errors have been made. After successful testing of the application, the command "compile\_iss()" can finally be executed. This provides the files in the selected directory "app\_dir" which are necessary for the installation of the programs. This process may take a few minutes.
2. After completion of the process, the app can be installed on all current Windows PCs. The corresponding installation file is located in the subfolder "RInno\_installer". As previously defined, the license texts and the default installation path should be displayed as expected.
3. After successful installation the app can be opened. The first time the application is executed, the loading time may be somewhat delayed due to the one-time loading of packages. However, this should no longer be the case in subsequent use.

# troubleshooting

In general, the occurrence of errors within the deployment process cannot be completely ruled out, even if the deployment process is carried out according to the instructions. Depending on the source of the error, its correction is trivial or requires special IT prior knowledge. In order to prevent such errors, it is therefore advisable to follow the above instructions as closely as possible.

In order to limit the errors, the general functionality of the application should first be checked using the "runApp" command of the deployment script. If an error message already appears here, an error seems to have occurred when changing the app parameters. Typical errors in this phase are:

* One special character too much / too little (e.g. . , / ( ) \* # )
* Wrong spelling of colors (see following link for Hexagon spelling)
* Wrong naming of . ico and . png files
* Updates within the loaded packages that lead to a change in the way individual functions are used.
* Other errors received by active modification & extension of the code

If such an error occurs, the associated error message marked in red within R should first be examined. If this does not immediately reveal the source of the error, it is advisable to enter all or part of it in a search engine and search for an explanation in forums. The "Stack Overflow" and "Github" platforms in particular have proved their worth. If the application is displayed normally, the raw application or its underlying code can most likely be classified as error-free.

For further troubleshooting the functionality of the deployment package "RInno" should be checked. It is used to convert the raw application (consisting of the two .R files "ui.R" and "server.R") into a classic installation file. Below is a list of possible sources of errors within the deployment process:

* Incorrect directory and / or folder specification
* "server.R" or "ui.R" are still open during the deployment process  
   (or still stored in the cache of an RStudio session)

The list does not claim to be complete, which means that further, unknown errors may occur in the future. Therefore it is recommended to check the [troubleshooting page of](https://github.com/ficonsulting/RInno/issues) "RInno" or the other "Package" authors if an unknown error occurs and to check for suitable solution options. If you do not find a satisfactory answer here, you should consider reporting the error. This only requires a free registration under Github and a few clicks under the tab "File an Issue". Authors themselves usually have a great interest in reporting errors and usually react to them very quickly and courteously.

# limitations

Due to the limited resources, the tool offers additional feature ideas that could not yet be integrated. These are listed below:

* Unfortunately, the approach only works for Windows operating systems, an offline solution for Mac operating systems is not available at this time.
* Automatic Feature Elimination: Slider to make a variable selection during modeling via Recursive Feature Elimination. Accordingly, validation options similar to the hyper parameter selection would have to be provided.
* Database override: Possibility to write the predicted values directly into a database in the Prediction Tool.

The support of collaborators in the creation of these extended functionalities is expressly desired. Further functionality suggestions can be made on the corresponding Github page.

# Concluding remarks:

* The dashboard contains appealing visualizations based on the Highcharts package. Commercial use of this library requires the purchase of a license from the [following link](https://shop.highsoft.com/).
* Despite careful checking for bugs and unintentional behaviour, there is no guarantee of freedom from errors. It is published under the Github platform and continuously optimized to eliminate sources of errors.