Expert Systems With Applications 213 (2023) 118912

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

The automation of the development of classification models and improvement of model quality using feature engineering techniques

Sjoerd Boeschoten [a], Cagatay Catal [b,*], Bedir Tekinerdogan [a,*], Arjen Lommen [c], Marco Blokland [c]

[a] Information Technology Group, Wageningen University & Research, Wageningen, The Netherlands
[b] Department of Computer Science and Engineering, Qatar University, Doha, Qatar
[c] Wageningen Food Safety Research Institute, Wageningen University & Research, Wageningen, The Netherlands

A R T I C L E I N F O

Keywords:
Machine learning pipeline
Feature engineering
Machine learning
Automation
Data imputation
Feature transformation
Data balancing

A B S T R A C T

Recently pipelines of machine learning-based classification models have become important to codify, orchestrate, and automate the workflow to produce an effective machine learning model. In this article, we propose a framework that combines feature engineering techniques such as data imputation, transformation, and class balancing to compare the performance of different prediction models and select the best final model based on predefined parameters. The proposed framework is extendable and configurable by adding algorithms supported by the CARET package implemented in the R programming language. This framework can generate different machine learning models, which provide comparable results compared to other studies. The framework allows practitioners and researchers to automatically generate different classification models. This research used High-Resolution Orbitrap-based Mass Spectrometers (HRMS) data to create automated prediction models for the first time in literature. We demonstrated the applicability of feature engineering techniques such as data imputation, transformation (e.g., scaling, centering, etc.), and data balancing using several case studies and the proposed semi-automated framework. We showed how the initial prediction models can be improved using the proposed framework.

1. Introduction

The use of growth promoters to increase muscle mass in cattle is forbidden within the European Union (Qaid & Abdoun, 2022). To control growth promotor abuse in cattle, samples are taken at farms and mainly measured using targeted methods (LC-triplequad-MS methods). A targeted method only detects a fixed predefined set of compounds. However, due to technical advances in the last years, it is also possible to obtain highly accurate data in a non-targeted manner using liquid chromatography high-resolution orbitrap-based mass spectrometers (LC-HRMS) (Bianco et al., 2022). Using LC-HRMS, many data-rich files are collected containing exact masses, intensities, and retention times. Elemental compositions (molecular formulas) are deduced from the exact masses. For each LC-HRMS file, all signals with the same elemental composition are binned by adding the intensities. The new binned dataset, therefore, consists of elemental compositions and summed intensities for each elemental composition. The data itself is sparse with low completeness regarding intensities. Therefore, data cleaning (Ilyas & Rekatsinas, 2022) and feature extraction methods are important parts of building machine learning models.

Machine learning pipelines (Topçuoğlu et al., 2021) include a number of steps from data extraction and preprocessing to model training and deployment. Traditionally, the machine learning steps were performed as a manual process. Recently, machine learning pipelines have become important to codify, orchestrate, and automate the workflow to produce effective machine learning models. In this article, we propose a framework that combines feature engineering techniques (Gibert et al., 2022) such as data imputation (Neves et al., 2022), transformation, and class balancing (Khatir & Bee, 2022) to compare the performance of different prediction models and select the best final model based on predefined parameters.

This study focuses on the current state of the art of feature engineering techniques, which are investigated, compared, and implemented to reach an optimal classifier model. Examples include missing

* Corresponding authors.
E-mail addresses: sjoerd.boeschoten@wur.nl (S. Boeschoten), ccatal@qu.edu.qa (C. Catal), bedir.tekinerdogan@wur.nl (B. Tekinerdogan), arjen.lommen@wur.nl (A. Lommen), marco.blokland@wur.nl (M. Blokland).

https://doi.org/10.1016/j.eswa.2022.118912
Received 27 May 2022; Received in revised form 12 September 2022; Accepted 23 September 2022
Available online 27 September 2022

data imputation, variable transformation, encoding, scaling, aggregating, dimension reduction, and feature creation/extraction/selection. These techniques are combined and tested with several machine learning models for robustness, efficacy, and ease of use purposes. To simplify the development of all these prediction models, a new framework is built on top of the CARET package by developing wrapper functions.

In addition to the new framework development aim, this study also aims to lead to a collection of models, capable classifiers for the usage to mine HRMS data files, and increase knowledge on important compound interactions and even better testing methods. For instance, if only certain specific compounds are causing most of the variance, it may be necessary to develop such a test instead of an all-out urine sample HRMS analysis on all compounds. Moreover, it is possible that different effects are visible in this data unrelated to illegal growth promoter treatment, which can be useful for other monitoring tasks.

To demonstrate the applicability of the new framework on different datasets, we have also utilized a publicly available dataset called Pima (Smith et al., 1988), which was retrieved from the National Institute of Diabetes and Digestive and Kidney Diseases. Using this dataset, researchers aim to predict whether a patient has diabetes or not based on certain measurements. As such, we performed our experiments on both the HRMS and Pima datasets and demonstrated that the proposed framework simplifies the development of machine learning models dramatically.

The contributions of this paper are as follows:

1. We designed and implemented a framework for automatically generating classification models: In this research, a new framework was built using the CARET package and several wrapper functions were developed as part of this implementation. The framework allows practitioners and researchers to automatically generate different classification models automatically. With the help of this framework, it is possible to find the most optimal classification model.
2. We created multiple classification models for the High-Resolution Orbitrap-based Mass Spectrometers (HRMS) data and aimed to find the best performing one: Finding the best performing classification model for the HRMS data requires a lot of effort and time. Therefore, many different models need to be built and investigated. In this research, these required models were built and the best performing one was identified. This kind of research has not been carried out for the HRMS dataset of urine profiles like this before. This type of model is very beneficial to controlling growth promotor abuse in the cattle industry and we were able to develop different classification models for this purpose.
3. We showcased the applicable feature engineering techniques within this framework: For automated prediction model creation, the framework allows different feature engineering techniques such as data imputation, transformation (e.g., scaling, centering, etc.), and data balancing. We demonstrated the applicability of these feature engineering techniques, which are crucial elements of machine learning models, in this semi-automated framework.
4. We demonstrated the ways to improve model quality using this automation framework: With the help of case studies, we were able to show how the initial prediction models can be improved using the proposed framework. Each technique might have a different effect on the overall performance of the model and therefore, we showed the performance change after a certain technique is applied.

The next sections are organized as follows. Section 2 describes the background and related work. Section 3 presents the research methodology. Section 4 discusses the framework that supports the creation of automated prediction models. Section 5 explains the evaluation of the framework on different datasets. Section 6 presents the discussion and threats to validity. Section 7 concludes the paper and shows the potential future work.

2. Background and related work

2.1. Liquid chromatography coupled to mass spectrometry

Liquid chromatography (LC) is a method to separate compounds based on interactions of the compounds with the adsorbent in the column. The time (retention time) compounds are retained on the column is dependent on the strength of their interactions. The detection of an eluting compound is often done with a mass spectrometer that is coupled to the LC (for example LC-Orbitrap-MS). A schematic example is shown in Fig. 1. As shown in this figure, the mass spectrometer involves the following three components: Ion Source, Mass Analyzer, and Detector. In the first component, the sample is ionized and cations are generated. Later, the second component separates ions based on their mass. Finally, the detector component detects the quantity and species of the ion. These steps create the main process performed by a mass spectrometer. Although a mass spectrometer is accurate for the detection and separation, it is not sufficient for a very complex mixture. Therefore, it is combined with High-Performance Liquid Chromatography (HPLC), and this combination is called Liquid Chromatography Mass Spectrometry (LC-MS). This combination of techniques provides better accuracy and is used in many different application domains.

Mass Spectrometry (MS) analysis is an analytical method to derive information on the presence and concentration of compounds based on the mass and charge of these compounds. It is widely used in organic chemistry applications and analysis varying from pharmaceutical analysis (Loos et al., n.d.) to biomolecule characterization (Breuker et al., 2008), environmental analysis (Petrovic et al., 2005), and forensic analysis (Hoffmann et al., 2008).

In this study, the HRMS dataset is derived in an LC-Orbitrap-MS setup. Compounds are separated by LC and subsequently ionized and passed on to the orbitrap. Ions in this Orbitrap oscillate at varying frequencies. Measuring these frequencies allows for the derivation of mass over charge (M/Z) and subsequently of mass spectra images using Fourier transformations. The Orbitrap analyzers allow for high mass accuracies with mass error < 5 ppm (parts per million of the mass). An example of the Orbitrap is shown in Fig. 2. Typical data consists of the time necessary for compounds to elute (retention time) from the LC column, their exact mass over charge ratios (i.e. from their ionized form), and their intensity. The elemental composition of the compound can be calculated from the exact mass.

2.2. Machine learning

Machine learning (ML) is a sub-field of artificial intelligence, which combines the computational power of computers with well-established statistical algorithms to accomplish a plethora of tasks (Jordan & Mitchell, 2015; Muhamedyev et al., 2015). These tasks can range from a simple regression of numerical variables to complicated neural networks classifying images, or detecting human speech in recorded audio.

The use of ML requires a workflow of first gathering and preparing the required data, analyzing which models should be used, and subsequently, training and evaluating the models based on evaluation metrics. In addition, hyperparameter optimization/hyperparameter tuning can drastically alter the model performance. Each of these steps shown in Fig. 3 involves different challenges. For instance, selecting a suitable algorithm to build the model has many different options, and also, preprocessing of the gathered data involves several techniques. In some cases, labeled data might not be sufficient to build a high-performance prediction model, and therefore, unlabeled data are also considered (i.e., semi-supervised learning). Each of these steps might take a considerable amount of time and recently, there is a research field called Automated Machine Learning (AutoML) that aims to provide techniques to make machine learning available for non-machine learning experts.

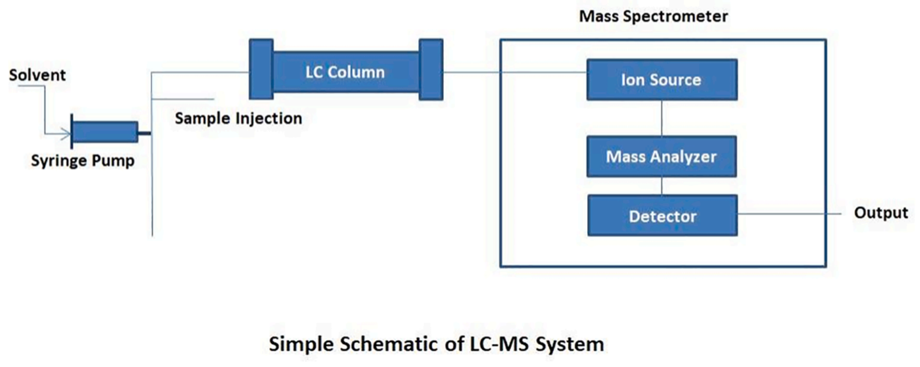

Fig. 1. Schematic overview of the experimental workflow of MS analyses (Chemyx, 2021).

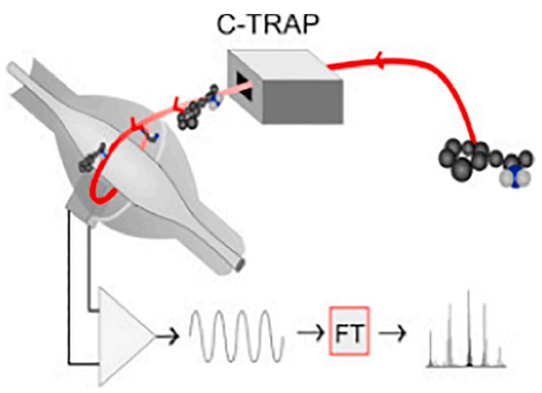

Fig. 2. Example of the derivation of mass/charge diagram based on the orbitrap *(*Rajawat & Jhingan, 2019*)*.

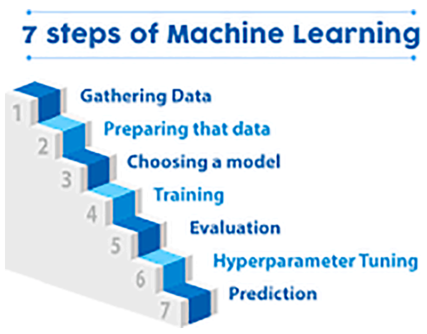

Fig. 3. Standard workflow of machine learning model development (Advani, 2021).

Since the success of the prediction models highly depends on the expertise of the machine learning experts, this research field aims to simplify the development of machine learning models. There are several packages such as AutoWEKA, Auto-sklearn, and H2O AutoML, which were developed for automated machine learning. In deep learning, there is also a similar research field called neural architecture search (NAS) that aims to find a well-performing architecture automatically. Current deep learning models are built using many different options (e.g., different layer types, activation functions, algorithms, etc.) and therefore, the development is time-consuming and requires a lot of effort from the deep learning experts. For deep learning, there are also some packages such as AutoKeras and Auto-PyTorch.

This workflow shown in Fig. 3 is often a manual search for the right combination of data preparation, model choice, and model tuning. This can be a time-consuming task, especially if certain model performance criteria must be met. There are packages such as CARET (Kuhn, 2008) for R, which help immensely with providing a framework for a data scientist to quickly experiment and build models. However, this is not fully automated and in order to quickly build well-performing models, this workflow should be automated.

2.3. *Feature engineering*

Feature engineering is often described as the art of extracting useful features or properties from data using either domain knowledge or established transformation methods (Heaton, n.d.). This is done by trying arbitrarily or with well-educated guesses to create new variables. Later, these new sets of features are used in the ML workflow to create a model, which is subsequently tested for model performance and feature importance.

Common ways of feature engineering are cleaning data, handling missing data, and transforming the data. An example of transforming data is simple numerical transformations such as log scaling to force more normal distributions in the dataset, clustering to group data that is alike, encoding data to create categorical data, and other methods that try to capture the raw data in a different way such as PCA (Principal Component Analysis) (Jollife & Cadima, 2016).

Missing data can be a problem in spare datasets with many variables with low frequency in entries. This can lead to models not functioning at all to dropping useful information in model creation. This problem can be addressed by imputing missing data, dropping data, or a combination of both. Imputing is an easy way to replace or fabricate new values. This can be done by simple arithmetical substitutions such as by the median or average to a random distribution based on known data characteristics to more complicated methods such as MICE (Multivariate Imputation by Chained Equations) (Azur et al., 2011; Van Buuren & Oudshoorn, 2000), which allows modeling missing data on other variables.

Feature selection is often required to prevent excess in variables if methods such as feature creation or combinations are used, which can lead to dimensionality issues. This can be fixed by dimension reduction methods such as regularization or using feature selection to drop out certain variables based on relative significance or contribution to variance in the data. There are advancements in the field of automated feature engineering using tree-like methods to create new features (multi-relational decision tree learning) (Rinkal & Aluvalu, 2014) to deep feature synthesis (Kanter & Veeramachaneni, 2015). Methods like these have also become commercially available recently to help data scientists drastically increase data exploration capabilities and to help novices quickly extract useful features for further use in the workflow (It's All About the Features – Reality AI, n.d.). These methods are greatly helpful as they are able to beat human teams in ML competitions more often.

2.4. *Summary of the previous studies and contributions*

In this research, one of our main objectives was to build a high-performance machine learning model that can predict the samples of guaranteed treated animals correctly. This is our first case study in this research. There are some research papers that applied machine learning to the HRMS data (Liebal et al., 2020; Bouwmeester et al., 2020), and a few researchers used machine learning for the quantification of hormones in the samples using HRMS data (Rocha et al., 2022; Benedetto et al., 2021a; Benedetto et al., 2021b). However, they did not mention the challenges that exist in our HRMS data such as class imbalance, and also, they did not aim to develop the prediction models using a framework as proposed in our research. Our focus was to detect the illegal hormone abuse in the cattle breeding industry in the Netherlands and there were several challenges in our dataset such as class imbalance and missing data points, which were not addressed in this context before. Moreover, we aimed to build prediction models using our semi-automated framework to demonstrate its applicability in a complex real-world problem. Therefore, our framework, HRMS dataset, and the prediction models built on top of the HRMS dataset are considered novel.

3. Research methodology

3.1. *Research questions*

The following research questions have been identified in this study:

- **RQ1** - What are the adopted feature engineering methods that are applicable to numerical data for classification problems?
- **RQ2** – How can we improve the quality of a given classification model with feature engineering methods?
- **RQ3** – To what extent can we automate the development of classification models?

The motivation for each research question is shown in Table 1.

To answer these research questions, we followed the following steps:

For answering RQ1, relevant knowledge sources on feature engineering were collected using a domain analysis technique. The results of this step are provided in Section 5.1. For answering RQ2 and RQ3, a framework for automated prediction model creation is implemented as presented in Section 4. Further using this framework we adopt a machine learning life cycle, in which we apply different feature engineering techniques. The results are presented in Section 5. The approach is evaluated by using two case studies, which are explained in Section 4. The case studies and the adopted datasets are presented in Section 3. The overall approach is discussed in section 6.

3.2. *3.2 HRMS data set of urine profiles*

The data is provided in CSV format where every urine sample has its own CSV file and identifier containing information on which compounds are found within that sample. Measurements for compounds are retention time, elemental composition, intensity, and weight. Information on the source of these urine sample files is stored in description CSV files. The experimental conditions are stored in a separate CSV file containing information on which treatment is used in samples originating from animal experiments. For every unique sample, the data of the compounds are aggregated with their respective intensities in mind. It is unclear from the raw data what isomer each compound exactly is. This results in that different isomers being categorized as the same compound. The final result is a data frame containing the urine samples as rows with isomer intensities, treatment, and sex labels as columns. The final data is ordered in 273 columns and 1241 rows. 1241 samples are divided into two categories: treated and untreated. All unknown samples were assumed to be untreated, which amounted to 1185 samples, with the rest 56 as confirmed treated samples. Sex data was limited, with only 192 samples being labeled, the majority being female (176). The data is sparse with multiple columns containing missing data. The range of missing data is shown in Fig. 4.

The data used for the framework is intensity data grouped by the same elemental composition. Each elemental composition has its own variable and column. Each entry has its own label concerning the status of the sample. An example of data is shown in Table 2.

3.3. *Dataset from the university of california at irvine (UCI)*

The PIMA dataset is retrieved from the National Institute of Diabetes and Digestive and Kidney Diseases and directly read into the framework. The features of this dataset are shown in Table 3.

The PIMA dataset contains information on kidney diseases of patients of Pima Indian heritage in an effort to create predictions based on diagnostic measures.

4. Framework for supporting automated prediction model creation

In this section, we present the framework that we have developed for automation prediction model creation. The framework is based on the CARET (Classification And Regression Training) package implemented in R. This package is a tool used by data scientists as it aggregates the most often used methods for data preparation, model training, and tuning for evaluation, which allows rapid model creation, optimization, and testing. It supports 238 different algorithms with often multiple optimization parameters per model. It also allows for preprocessing, such as basic feature engineering and resampling to creating predetermined data splits for validation and optimization with, for example, cross-validation. In addition, it is possible to implement

Table 1
Overview of Research Questions.

Research Question	Motivation	Section
What are the adopted feature engineering methods that are applicable to numerical data for classification problems?	To establish well-performing workable flows of operations for repetition in other classification cases	5.1
How can we improve the quality of a given classification model with feature engineering methods?	To reach better performance evaluation scores from raw data	5.2
To what extent can we automate the development of classification models?	To ease model development and rapid deployment in practical cases	5.3

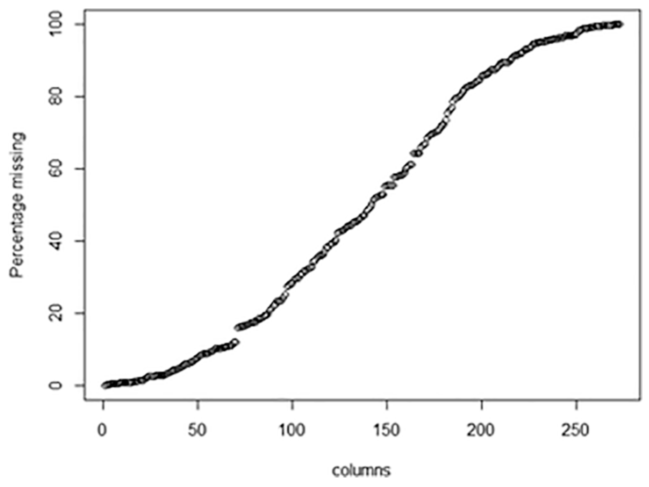

Fig. 4. Distribution of missing data.

Table 2
Example data of HRMS data set of urine profiles.

Sex	Label	C18H32O3	C18H24O8S2	… (more elemental compositions)
M/F/NA (factor)	bovine/ bovine test (factor)	NA/ 27930.88 (Double)	NA/ (Double)	

Table 3
Features of the PIMA dataset.

Feature Label	Type
# of times pregnant	Int
Plasma glucose concentration	Real
Diastolic blood pressure	Real
Triceps skin fold thickness	Real
2-hours serum insulin	Real
BMI	Real
Diabetes pedigree function	Real
Age	Int
Class	Binary

custom models or recipes, which allows for expansion in this system (12 Using Recipes with Train | The Caret Package, n.d.).

This study uses the CARET package to utilize most of the above-mentioned features in a semi-automatic way to create multiple models based on different data preparation and tuning methods. A diagram depicting this workflow is shown in Fig. 5, which depicts the order of operations from data imputation to transformations, which are then subsequently split into a validation set and a combined training/testing set, which is used for internal cross-validation for the hyperparameter tuning. The final performance is calculated using F1 scores based on unseen validation data to account for class imbalances.

4.1. Imputation

Missing data has to be addressed before training the model. Multiple ways of doing this are available. These vary from ignoring incomplete cases to replacing or calculating a distribution of values for the missing values. This can be done by simple methods such as a direct replacement or a normal or uniform distribution in a certain range to more advanced imputation methods. For example, knnImpute (Crookston & Finley, 2008) or MICE (Van Buuren & Oudshoorn, 2000), models the missing data based on other information stored in other variables by respectively applying a k-nearest neighbors algorithm or applying a chained regression model. In the HRMS data set of urine profiles, ignoring the incomplete cases is not an option as all entries would be dropped that way. Therefore, an imputation method must be chosen. An applied or hybrid strategy should be adopted by ignoring sparse variables and imputing the other variables. One easy way, given the fact that a lower detection limit is known, is to calculate a random distribution around this limit and use that as a lower bound. For the aggregated intensity data, this is not a big issue as the lower detection limit is multiple orders of magnitudes lower than most of the data. Therefore, this was chosen as the first approach.

4.2. Transformation

CARET allows for multiple transformation methods to be applied to the data such as scaling, centering, or calculated derivatives such as correlation scores and PCA, dropping zero variance (zv), near zero variance (nzv) columns. These transformations are immediately applied to all of the data points after imputation and before any split and class balancing occurs. Custom transformation methods can be used as well, however, these have to be implemented manually, which was done for the log transformation, which is not natively supported by CARET. Other methods could be implemented using CARET recipes.

4.3. Class balancing

This study deals with highly imbalanced data, which can affect model performance severely. Class imbalances can be addressed in multiple ways. This can be done by choosing models that can handle these differences relatively well or by applying the subsampling approach. Subsampling is often used to implement solutions to remediate model performance issues related to class imbalances. There are multiple options, down and upsampling, which respectively drop the frequency of the majority class or increase the frequency of the minority class. Smarter solutions are SMOTE (Chawla et al., 2002) or ROSE (Lunardon et al., 2014), which can do both to narrow the gap between the class imbalances by dropping samples and creating new ones based on others imputed by, for example, a k-nearest neighbor algorithm in the case of SMOTE. These methods are available as R packages. In this study, upsampling and SMOTE are used in conjunction with no modification to the training data. This means that the final validation set does not accurately represent the training data but does allow the learner models to get more information on the lower frequency class relative to the higher frequent class.

4.4. Training and tuning

The training uses the CARET package as a framework for both implementing models and tuning the hyperparameters for these learners. Every algorithm provided in the CARET package has standard suggested tuning grids for parameter optimization, which is used to find an optimal model within these parameter ranges. For this optimization, model performance is necessary to assert model quality. In this case, a stratified 10-fold cross-validation and 5 repeats are applied in the tuning

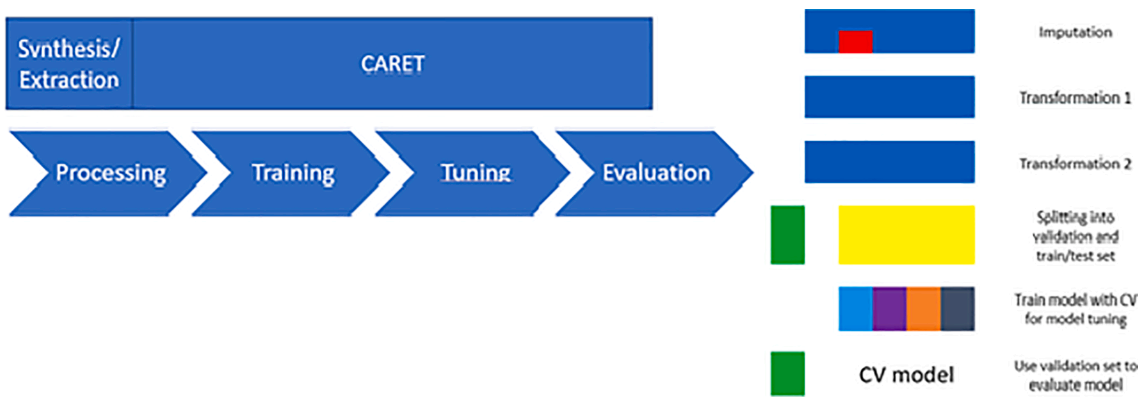
Synthesis/ Extraction
CARET
Processing
Training
Tuning
Evaluation
Imputation
Transformation 1
Transformation 2
Splitting into validation and train/test set
Train model with CV for model tuning
CV model
Use validation set to evaluate model

Fig. 5. Model Creation Process of the Framework.

process by measuring the overall misclassification error. Tuning is controlled by the standard suggested tune grid, which can be modified by changing the tuned length. This variable multiplies the grid components to control deeper or shallower searches for an optimal model. The main trade-off in this regard is time vs model quality. Larger tune lengths cost more time for a theoretically higher likelihood of better models. This tune grid can be adjusted manually per individual model to allow for time management. This is especially necessary for models that take significantly longer times, such as the gradient boosting methods. Different metrics can be applied by using the Mlmetrics package (Yan & Maintainer, 2016), which provides precision, recall, and F1 scores as evaluation metrics. F1 is calculated as shown in Equation 1. Moreover, it is possible to adjust the F score evaluation system by weighting the importance of precision and recall, as shown in equation 2 by changing β which allows for weighing for more importance of precision or recall.

$$F1 = 2*(precision*recall)/(precision + recall)(1)$$

$$F\beta = ((1 + \beta\text{^}2)*(precision*recall))/([(\beta]\text{^}2*precision + recall))(2)$$

This can be done in situations where a large class imbalance is present. This causes likely high recall but low precision for the learners used. Table 4 shows the algorithms, their descriptions, and the hyperparameters applied in this study. These were selected for ease of use and compatibility with the developed framework.

4.5. Validation

Final validation is performed with the final model produced by tuning in CARET on unseen and unmodified data with corresponding labels. For this, several split ratios are used to test the differences in sizes of the final validation set and the combined training/testing sets. The final model is applied to the unseen data and produces predictions, which are compared to the actual labels provided in the same dataset in a confusion matrix. From this table, metrics such as precision, recall, and F1 scores are calculated. This F1 score is calculated in the first place on the validation data. The model is then stored along its validation metrics and the corresponding datasets for possible further analysis depending on the F1 score, which is used as a first indicator and main selector for model quality. This workflow generates a multitude of final models and a much larger array of tuning models because each combination of imputation method, split ratio, preprocessing method, and modeling method is tested. This results in several hundreds of models due to the combinations. Saved models can be loaded and investigated for their hyperparameters and reused with any dataset related to the training set. This way performance on seen and unseen data can be assessed. Moreover, for every experiment and variation, it is possible to retrieve a distribution of model performance, as such, comparisons can be made between different split ratios, model methods, imputation methods, and class balancing approach.

Table 4
Hyperparameters for CARET-supported algorithms.

Algorithm name	Description	Hyperparameters
LMT	Logistic Model Trees	iter
Logitboost	Boosted Logistic Regression	nIter
xgbLinear	eXtreme Gradient Boosting	nrounds, lambda, alpha, eta
xgbtree	eXtreme Gradient Boosting	nrounds, max_depth, eta, gamma, colsample_bytree, min_child_weight, subsample
deepboost		num_iter, tree_depth, beta, lambda, loss_type
Naïve_bayes		laplace, usekernel, adjust
rocc	ROC-Based Classifier	xgenes

5. Results

5.1. Currently adopted feature engineering methods (RQ1)

RQ1 - What are the adopted feature engineering methods that are applicable to numerical data for classification problems?

The literature covers feature engineering often as an art. The predictive power of machine learning models is aimed to be improved by extracting information or features from a dataset and utilizing domain knowledge or other commonly used transformations. This has been a manual process up until recently. New methods and automated feature extraction tools such as deep feature synthesis provided automated discovery of features with the help of open source tools such as *Featuretools* (https://www.featuretools.com/). Common ways of feature engineering are cleaning data, handling missing data, and transforming the data. An example of transforming data is simple numerical transformations such as log scaling to force more normal distributions in the dataset, clustering to group data that is alike, encoding data to create categorical data, and other methods that capture the raw data in a different way such as PCA. Missing data can be a problem in very sparse datasets with many variables with low frequency in entries. This can lead to models not functioning at all or dropping useful information in model creation. This problem can be addressed by imputing missing data, dropping data, or a combination of both. Imputing is an easy way to replace or fabricate new values. This can be done by simple arithmetical substitutions such as by the median or average to a random distribution based on known data characteristics to more complicated methods such as MICE, which allows for modeling missing data on other variables.

Feature extraction creates new features from raw data based on an algorithm or predefined ruleset. Feature selection is often required to prevent excess in variables if methods such as feature creation or combinations are used, which can lead to dimensionality issues. CARET provides methods to alter the class proportions in the data by either down and upsampling or using hybrid methods such as SMOTE (Synthetic Minority Oversampling Technique) or ROSE (Random Over-Sampling Examples). These methods are applied to enable machine learning models to generalize better over the classes themselves at the cost of increased bias. It is not always necessary to use such methods if the data is already well-balanced or if the minority class count is satisfactory. When such methods are used, special care needs to be taken with interpretation as some metrics do not reflect that the model performs better. An example of this is the accuracy paradox where it seems that accuracy increases while the model itself only predicts one class. Therefore, it is important to account for performance across classes and which metrics such as precision, recall, and F1 scores are to be used. SMOTE is used in this framework as the method for addressing the class balance issues in the HRMS data set of urine profiles as the positive label count was low and several configurations were tested and compared in Section 5.2. The final F1 scores were calculated on validation data that was not treated with the class balancing approach to best reflect the reality of the naturally found class distribution.

5.2. Improving classification models with feature engineering methods (RQ2)

RQ2 – How can we improve the quality of a given classification model with feature engineering methods?

Several experiments were run using different algorithms and transformation methods, as shown in Table 5. These combinations of imputation, transformation, and modeling methods were all then performed by the framework with the models, standard *tunegrid* was used as arguments for the hyperparameter optimization unless stated otherwise.

Combining these methods with the fact that two different split ratios were used for the final validation set results in 6 (transformation methods) * 6 (model algorithms) * 2 (split ratios) = 72 models that were

Table 5
Algorithms and transformation methods.

Model algorithms	Transformation methods
deepboost	center
LogitBoost	corr
Naive_bayes	log
rocc	nzv
xgbLinear	scale
xgbTree	zv

created in the framework per experiment. However, each of these models is the final model based on the fine-tuning performed by CARET, therefore, essentially multiple hundreds of models are created by the framework based on the provided *tunegrids*.

Every experiment contains 72 final models. However, as there is a large class imbalance, multiple experiments were run with differing approaches to address this problem. These are shown in Table 6. Every experiment containing 72 final models is then loaded back into the framework for validation based on the separated validation set. Every model is then back-tested and scored with the F1 score. This results in 72 F1 scores for every experiment, which is used for comparisons and graphing. Moreover, the F1 scores are split by their respective modeling and transformation methods based on the average across the two split ratios resulting in tables with 36 averaged F1 scores along their respective precision and recall for comparison between methods and experiments.

Fig. 6 shows the model F1 performance distribution for the standard balanced dataset. The best model has an F1 score of 0.833 and is shown as the single bar on the right. This distribution is skewed to lower-performing models with 23/72 models performing above 0.5 F1 scores. Fig. 7 shows model performance metrics per algorithm and transformation for non-balanced data.

The model performance values are also shown in Tables 7–9 averaging on the split ratios of 0.8 and 0.9. The F1 score is also split into the precision and recall counterparts. Simple log transformation performs the best in combination with the Naive Bayes and ROC-based classifier. As shown in Table 8, the precision value of the Naïve Bayes model is 0.78 when the log transformation is applied. According to Tables 8 and 9, Logitboost algorithm provided 0.68 precision and 0.37 recall values.

Some transformation methods are completely biased towards one class indicating an F1 score of 0 or that specific algorithm resulted in an error also indicating a score of 0. Logitboost followed by the xgb algorithms is more consistent based on the transformation method.

Table 7 shows the final optimized model F1 score for every classification algorithm and transformation method in this study averaged for the two split ratios.

Table 8 shows the final optimized model precision for every classification algorithm and transformation method in this study averaged for the two split ratios.

Table 9 shows the final optimized model recall for every classification algorithm and transformation method in this study averaged for the two split ratios.

Fig. 8 shows the model F1 performance distribution for the standard Smote balanced dataset. The best model has an F1 score of 0.923 and is shown as the single bar on the right. This distribution is skewed to higher performing models indicating class balancing shows a general improvement. These model performance results are also shown in Tables 11 to 12 averaging on the split ratios of 0.8 and 0.9. The F1 score in

Table 6
Experiments with subsampling.

ID	Additional conditions
1	No subsampling
2	Standard SMOTE (200 % max upsampling)
3	SMOTE (400 % max upsampling)

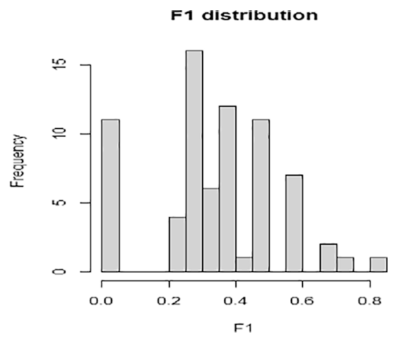

Fig. 6. F1 distribution with no subsampling (highest F1 validation = 0.833).

Table 10 is also split into the precision and recall counterparts in Table 11 and 12 respectively. The tables are also depicted as a bar chart in Fig. 9. All models tend to dominate higher recall vs lower precision, which tends to provide more false positives. As shown in Tables 11 and 12, xgbLinear provided 0.62 precision (i.e., log transformation) and 0.96 recall values.

Gradient boosting performs better with the linear variant being on top. Overall, correlation transformation is the best single transformation step followed by log, centering, matrix, and scaling. The worst performing models are Naive Bayes and the Rocc algorithm with the exception of log transformation for Naive Bayes.

Table 10 shows the final optimized model F1 score for every classification algorithm and transformation method in this study averaged for the two split ratios.

Table 11 shows the final optimized model precision for every classification algorithm and transformation method in this study averaged for the two split ratios.

Table 12 shows the final optimized model recall for every classification algorithm and transformation method in this study averaged for the two split ratios.

Fig. 10 shows the model F1 performance distribution for the Smote 400 % balanced dataset. The best model has an F1 score of 0.857 and is shown as the single bar on the right. This distribution is divided into three lower to mediocre sections and higher performing models with roughly the same prevalence. These model performance values are also shown in Tables 13 to 15 averaging on the split ratios of 0.8 and 0.9. The F1 score in Table 13 is also split into the precision and recall counterparts in Table 14 and 15 respectively. The tables are also depicted as bar charts in Fig. 11. All models tend to dominate higher recall vs lower precision, which tends to more false positives.

As shown in Tables 14 and 15, deepboost algorithm provided 0.80 recall (i.e., nzv transformation is applied) and 0.65 precision values.

Deepboost is at the top closely along with the xgb linear and the log-transformed Naive Bayes classifier. Overall, log transformation is the best single transformation step followed by correlation. The worst-performing models, in general, are Naive Bayes and the Rocc algorithm.

Table 13 shows the final optimized model F1 score for every classification algorithm and transformation method in this study averaged for the two split ratios.

Table 14 shows the final optimized model precision for every classification algorithm and transformation method in this study averaged for the two split ratios.

Table 15 shows the final optimized model recall for every classification algorithm and transformation method in this study averaged for the two split ratios.

The Pima dataset has a range of F1 scores from the lower 0.55 to the upper 0.7 with 10 finalized and optimized models being in the highest section. This framework is able to create models with similar performance as in literature (Pal et al., 2020). This paper cites a maximum F1 score of 0.64 on an SVM classifier after balancing the data. The results in this section were not generated after balancing the Pima data. This

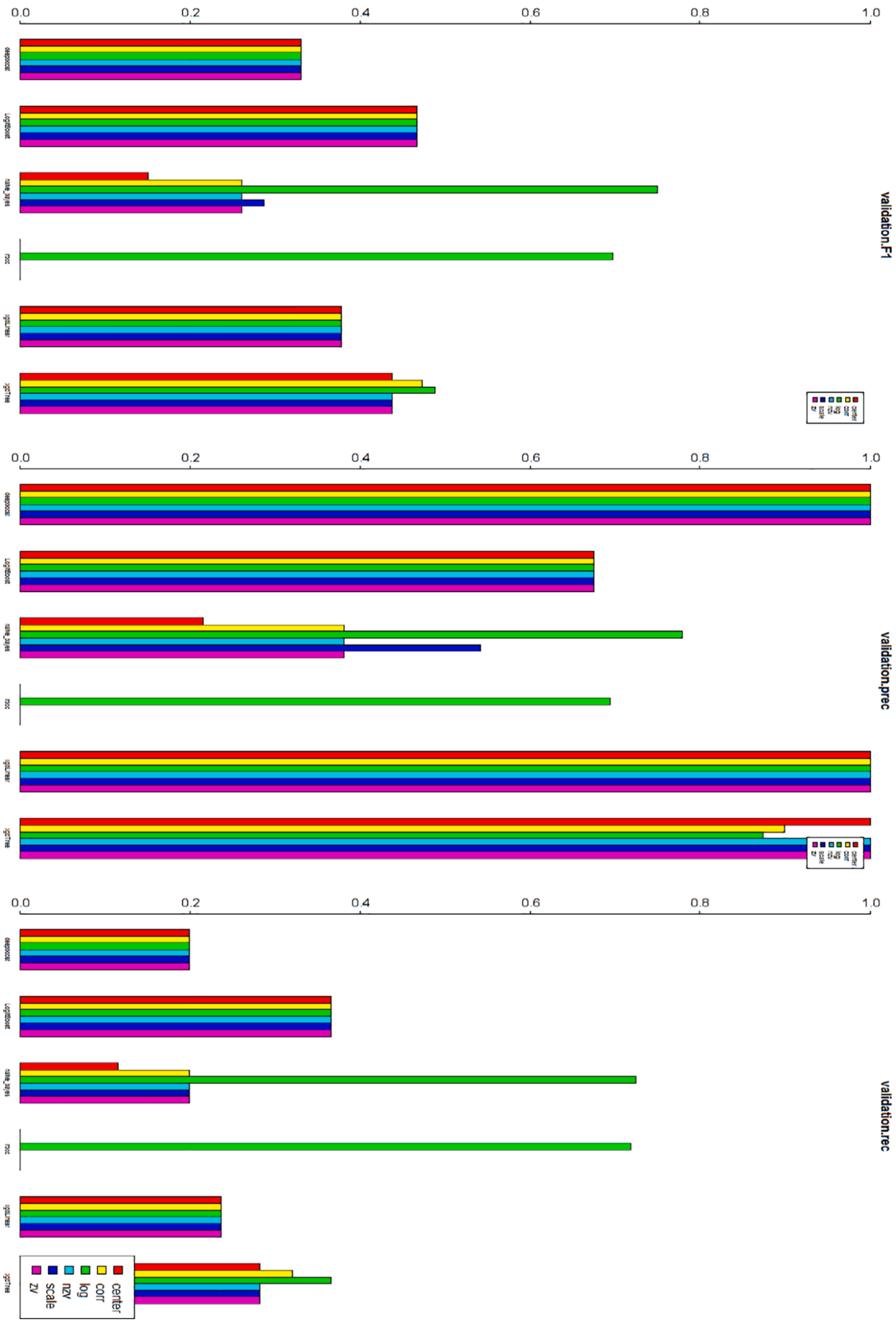

Fig. 7. Model performance metrics per algorithm and transformation for non-balanced data.

Table 7
Validation F1 averaged on the split ratio.

	center	corr	log	nzv	scale	zv
deepboost	0.33	0.33	0.33	0.33	0.33	0.33
LogitBoost	0.47	0.47	0.47	0.47	0.47	0.47
naive_bayes	0.15	0.26	0.75	0.26	0.29	0.26
rocc	0	0	0.7	0	0	0
xgbLinear	0.38	0.38	0.38	0.38	0.38	0.38
xgbTree	0.44	0.47	0.49	0.44	0.44	0.44

Table 8
The precision average on the split ratio.

	center	corr	log	nzv	scale	zv
deepboost	1	1	1	1	1	1
LogitBoost	0.68	0.68	0.68	0.68	0.68	0.68
naive_bayes	0.21	0.38	0.78	0.38	0.54	0.38
rocc	0	0	0.69	0	0	0
xgbLinear	1	1	1	1	1	1
xgbTree	1	0.9	0.88	1	1	1

indicates that this framework is able to create at least similar models as to what an individual could do.

Fig. 12 shows the model F1 performance distribution for the Pima diabetes dataset. These model performance values are also shown in Tables 16–18 averaging on the split ratios of 0.8 and 0.9. The F1 score in Table 16 is also split into the precision and recall counterparts in Tables 17 and 18 respectively. The tables are also depicted as a bar chart in Fig. 13. All models tend to dominate higher precision vs lower recall

Table 9
The Recall average on the split ratio.

	center	corr	log	nzv	scale	zv
deepboost	0.2	0.2	0.2	0.2	0.2	0.2
LogitBoost	0.37	0.37	0.37	0.37	0.37	0.37
naive_bayes	0.12	0.2	0.72	0.2	0.2	0.2
rocc	0	0	0.72	0	0	0
xgbLinear	0.24	0.24	0.24	0.24	0.24	0.24
xgbTree	0.28	0.32	0.37	0.28	0.28	0.28

Table 10
Validation F1 averaged on the split ratio.

	center	corr	log	nzv	scale	zv
deepboost	0.62	0.66	0.55	0.62	0.62	0.62
LogitBoost	0.45	0.45	0.5	0.45	0.45	0.45
naive_bayes	0.21	0.21	0.61	0.21	0.21	0.21
rocc	0.29	0.29	0.45	0.29	0.37	0.29
xgbLinear	0.71	0.77	0.75	0.71	0.71	0.71
xgbTree	0.59	0.61	0.66	0.59	0.59	0.59

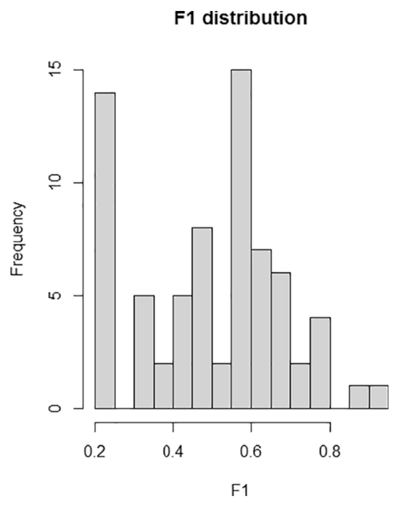

Fig. 8. Distribution for standard Smote balancing (200 %) (standard, highest validation F1 = 0.923).

Table 11
The precision average on the split ratio.

	center	corr	log	nzv	scale	zv
deepboost	0.46	0.5	0.4	0.46	0.46	0.46
LogitBoost	0.3	0.3	0.34	0.3	0.3	0.3
naive_bayes	0.12	0.12	0.51	0.12	0.12	0.12
rocc	0.17	0.17	0.3	0.17	0.24	0.17
xgbLinear	0.58	0.67	0.62	0.58	0.58	0.58
xgbTree	0.46	0.45	0.51	0.46	0.46	0.46

Table 12
The Recall average on the split ratio.

	center	corr	log	nzv	scale	zv
deepboost	0.96	0.96	0.88	0.96	0.96	0.96
LogitBoost	0.92	0.92	0.92	0.92	0.92	0.92
naive_bayes	0.88	0.88	0.76	0.88	0.88	0.88
rocc	0.88	0.88	0.96	0.88	0.88	0.88
xgbLinear	0.92	0.92	0.96	0.92	0.92	0.92
xgbTree	0.84	0.92	0.96	0.84	0.84	0.84

although the difference is smaller than in the HRMS data set of urine profiles.

As shown in Tables 17 and 18, xgbLinear provided 0.73 precision and 0.62 recall values.

xgbLinear is at the top closely along with the xgbtree and deepboost. Overall, log transformation is the best single transformation step. The worst-performing models, in general, are Naive Bayes and the Rocc algorithm. The log transformation is the best for Naive Bayes although the gain is smaller than for the HRMS data set of urine profiles.

5.3. *Automation of classification models (RQ3)*

RQ3 – To what extent can we automate the development of classification models?

While there are methods and algorithms well suited for automatic classification learning such as deep learning models, it is not often that the entire pipeline from data loading to the final classification model is automated. A data scientist regularly has to adjust model learning parameters or try different feature engineering methods in an effort to improve model quality. Parts of this pipeline are available to be automated such as automatic feature extraction or automated model tuning. Examples of this are the Multi-Relational Decision Tree Algorithm (MRDTL) (Atramentov et al., 2003) or the aforementioned Deep Feature synthesis or CARET itself for automated model tuning. Ideally, the entire pipeline as shown in Fig. 14 needs to be automated.

The framework as explained in Section 4 focuses on the data preparation and model building and training sections with focuses on data transformation and automated model testing and validation. Hyperparameter tuning is also automated but is included in the CARET package and not edited for this study with the exception of the ability to change tunegrid length or provide a custom tune grid. Results shown in Section 5.2 were generated by this framework. Each model is individually constructed and saved as an rds file. Afterward, these were loaded and the results were analyzed and compiled for automatic generation of the distribution graphs and tables. The parts of creating and loading the models are separated resulting in few actions required to generate these graphs. These are manually checking the data for the requirements of having one label class and the rest as numerical, designating the label column, setting parameters for model creation such as vectors of which algorithms, transformations, tunegrids to use, and running the model creation parts, running the model analysis and plotter part. These actions themselves can be performed within a few minutes. The system is, therefore, not fully automatic but does achieve automation of model creation including the tedious parts of hyperparameter optimization. The framework has proven to be working with two separately sourced datasets.

The framework as explained in Section 4 focuses on the data preparation and model building and training sections and covers the data transformation and automated model testing and validation. Hyperparameter tuning is also automated but is included in the CARET package and not edited for this study with the exception of the ability to change tunegrid length or provide a custom tune grid.

6. Discussion

6.1. *General discussion*

The development of machine learning models is shown to be automated as long as a strict frame is defined on what to support. For this, basic building blocks of model development were investigated and implemented using two case studies. There are, however, many more methods available for more niche cases. This would increase complexity in terms of time and implementation as combinations of these building blocks can grow quickly. They can be implemented due to the recipe function of CARET, which allows this framework to be customized for special cases. This framework is able to quickly generate an array of

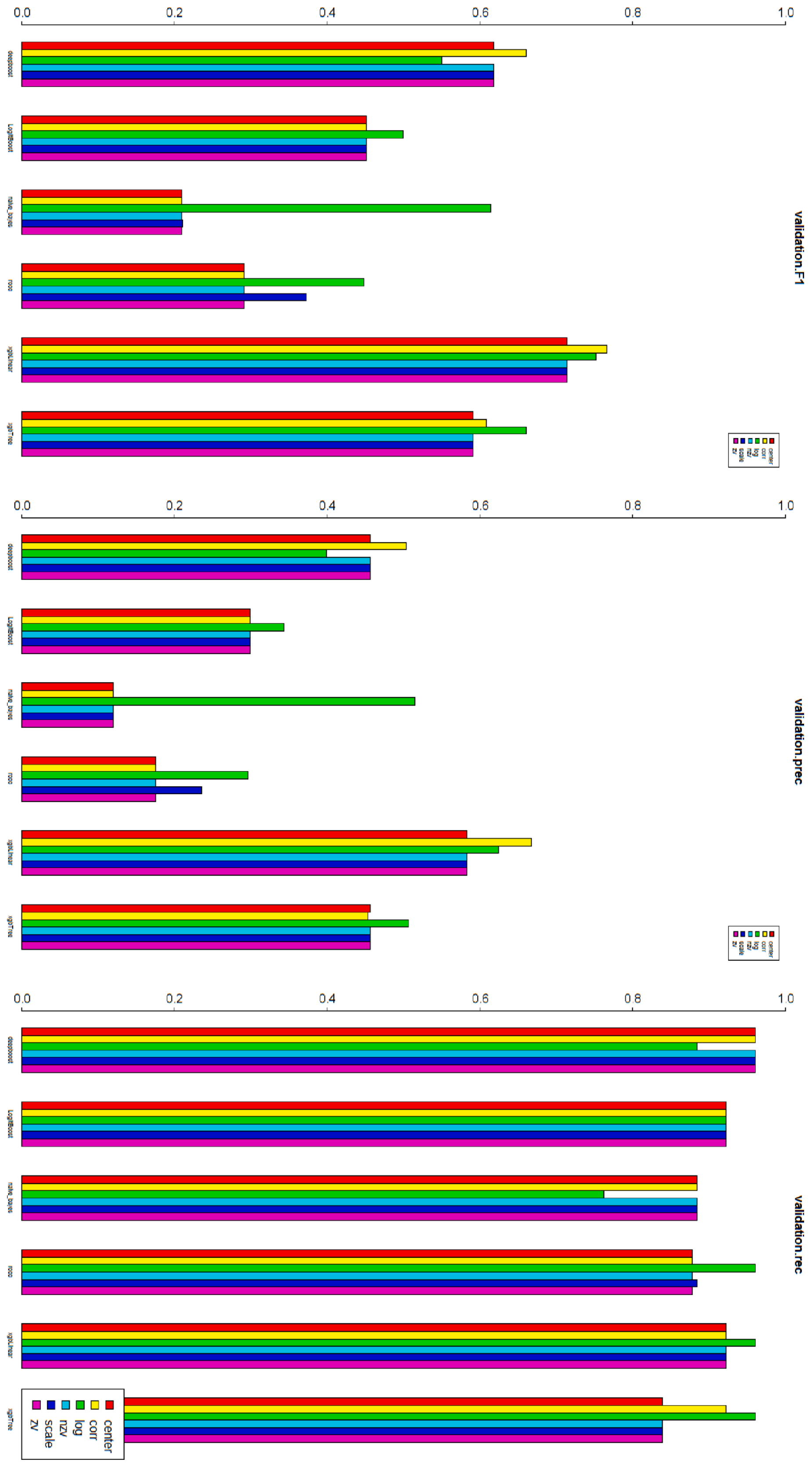

Fig. 9. Model performance metrics per algorithm and transformation for standard Smote class balancing.

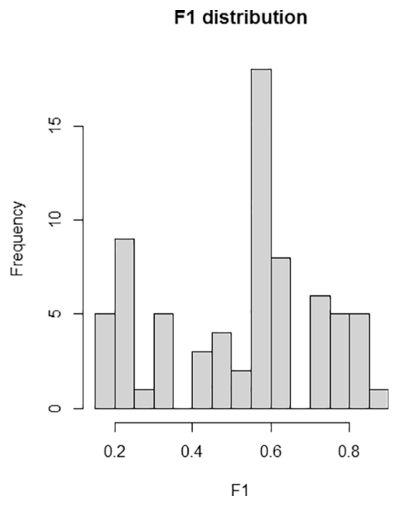

Fig. 10. Distribution for smote with 400 % oversampling (400 perc over, highest validation, F1 = 0.857).

Table 13
Validation F1 averaged on the split ratio.

	center	corr	log	nzv	scale	zv
deepboost	0.71	0.63	0.68	0.71	0.71	0.71
LogitBoost	0.52	0.56	0.49	0.52	0.52	0.52
naive_bayes	0.21	0.21	0.72	0.21	0.21	0.21
rocc	0.28	0.28	0.47	0.28	0.36	0.28
xgbLinear	0.72	0.72	0.74	0.72	0.72	0.72
xgbTree	0.58	0.69	0.69	0.58	0.58	0.58

Table 14
The precision average on the split ratio.

	center	corr	log	nzv	scale	zv
deepboost	0.65	0.64	0.55	0.65	0.65	0.65
LogitBoost	0.39	0.42	0.35	0.39	0.39	0.39
naive_bayes	0.12	0.12	0.72	0.12	0.12	0.12
rocc	0.17	0.17	0.32	0.17	0.23	0.17
xgbLinear	0.67	0.66	0.68	0.67	0.67	0.67
xgbTree	0.51	0.6	0.58	0.51	0.51	0.51

Table 15
Recall averaged on the split ratio.

	center	corr	log	nzv	scale	zv
deepboost	0.8	0.64	0.88	0.8	0.8	0.8
LogitBoost	0.81	0.85	0.84	0.81	0.81	0.81
naive_bayes	0.85	0.85	0.72	0.85	0.85	0.85
rocc	0.79	0.79	0.88	0.79	0.88	0.79
xgbLinear	0.81	0.81	0.85	0.81	0.81	0.81
xgbTree	0.77	0.81	0.85	0.77	0.77	0.77

models for different datasets with comparable performance results for the Pima dataset. The framework shows which models and transformation methods outperform each other and can be used to select a model and its defined workflow for a practical application.

RQ1 - What are the adopted feature engineering methods that are applicable to numerical data for classification problems?

A multitude of options regarding feature engineering exists. In creating any machine learning pipeline, it is necessary to decide which techniques should be adopted. The traditional way of trial and error is often employed by newbie data scientists. The process consists of creating or transforming features based on domain knowledge and subsequently, testing them in iterations, with each iteration hopefully increasing model performance or better alignment with the end goal. This is an iterative and time-consuming process and is reported to be the largest time-consuming task for data scientists. Choosing and testing these techniques becomes more challenging if domain knowledge is not readily available. In this case, the process becomes a stepwise iteration of randomly selecting a subset of these techniques and testing them. This study focuses on this method by automating this process based on some common and easy-to-understand transformations. This is done by testing combinations of different models for rapid model development and testing.

The HRMS data set of urine profiles entailed numerical transformations, principal component analysis (PCA), and filtering for near or non-zero variance features. For the HRMS data set of urine profiles, data were grouped based on chemical element composition based on the knowledge that certain carbon counts were expected to be an important feature in testing hormone treatment status. This data could have been grouped better by dividing these compositions into different compound classes to better reflect the chemical reality and perhaps to increase resolution. This was, however, not performed as it would increase the sparsity of the data greatly and the expectation that carbon classed compositions are somewhat alike in the pathway in bovines.

In theory, much more specific features could be synthesized based on biochemical knowledge such as dividing into acids, large molecules, intermediates, etc. This, however, requires specific domain knowledge and would exacerbate the missing data more. Therefore, the initial data grouping was kept simple as it was the goal to create a framework that would be able to perform on raw numerical data with limited knowledge and limited to no manual crafting of features. This framework investigates some simple transformations, which work on any numerical data for testing. Another reason to use this type of feature engineering was that other types such as encoding, aggregating, and extraction require manual intervention or would quickly increase the possibilities without an intelligent approach, which probably has to be based on domain knowledge. This framework, however, does not only do the feature engineering part of the machine learning pipeline but also addresses hyperparameter optimization and validation. It is known that sophisticated feature engineering approaches are available specifically for one field of feature engineering fields such as deep feature synthesis and MRDTL. Ideally, a perfect pipeline could combine the best of every step and use such methods as well at the cost of more time and complexity. This framework uses common and simple transformations to keep this process relatively simple. In this case, they were numerical transformations and imputations, PCA, and removal of zero or near zero variance variables. It is possible to include other transformations such as YeoJohson or to implement novel transformations if not supported by CARET. The latter requires a manual definition according to CARET's recipes.

RQ2 – How can the quality of a given classification model be improved with feature engineering methods?

The quality of models can both be increased as well as decreased by feature engineering in a step-by-step process. This is often described as an art in data science competitions. It is hard to find an optimal way of feature engineering as many approaches and combinations are available. However, domain knowledge makes this easier as it can act as a guide on how the data is supposed to be interpreted by models, for example, by performing a log distribution on biological data to force a normal distribution, which is often expected or by grouping and aggregating data such as compounds with the same elemental composition. In order to improve the quality as defined, multiple approaches must be tested and a satisfactory approach needs to be selected. It is, however, important not to over-engineer the data or overfitting can occur on the validation data if this is kept constant. For example, validation data is kept as a reality check for model performance trained only on the training data. However, if a large number of models is constructed and selected for performance on validation data, at some point the process also fits for the validation data specifically. It is therefore

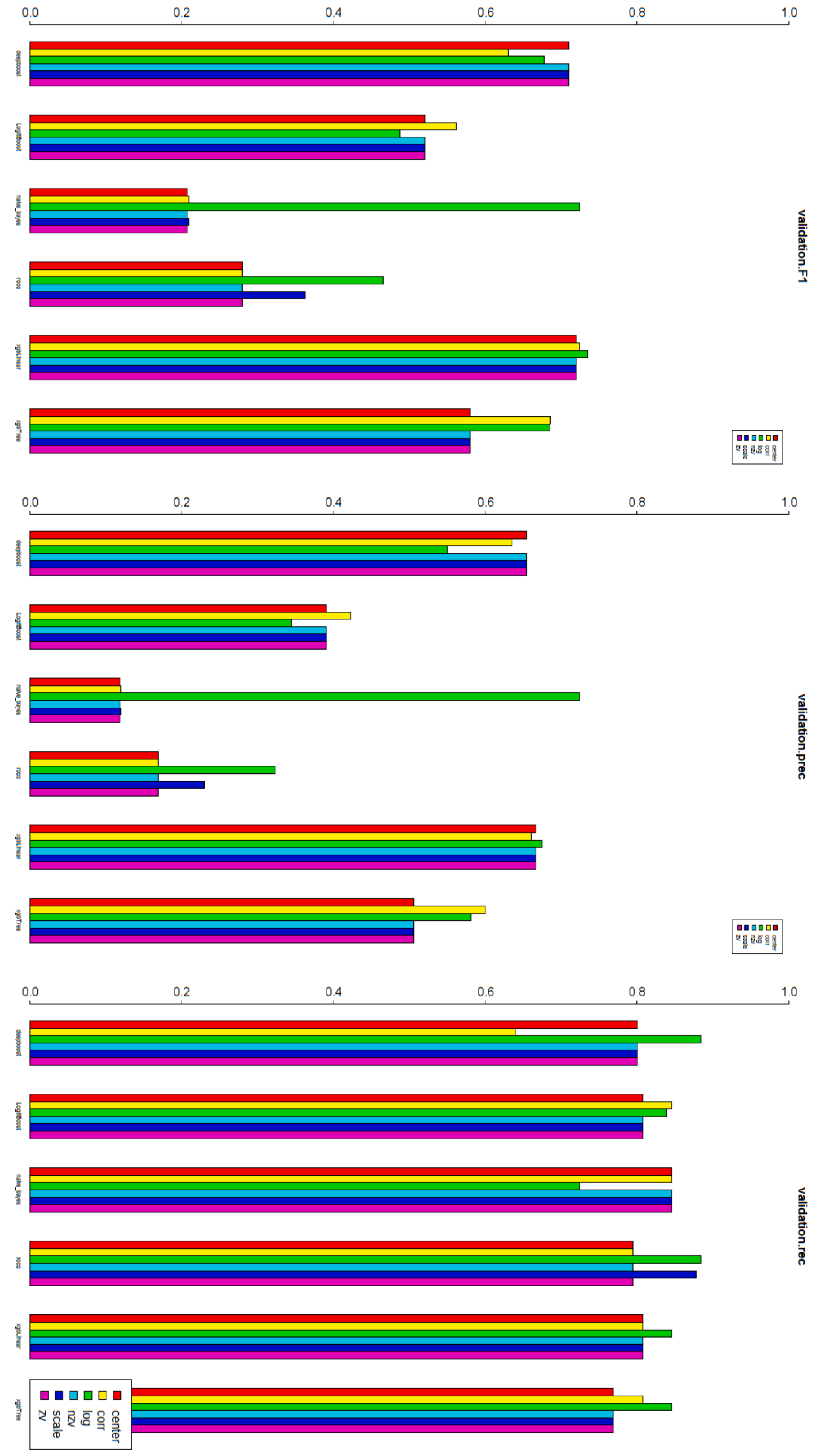

Fig. 11. Model performance metrics per algorithm and transformation for Smote 400% class balancing.

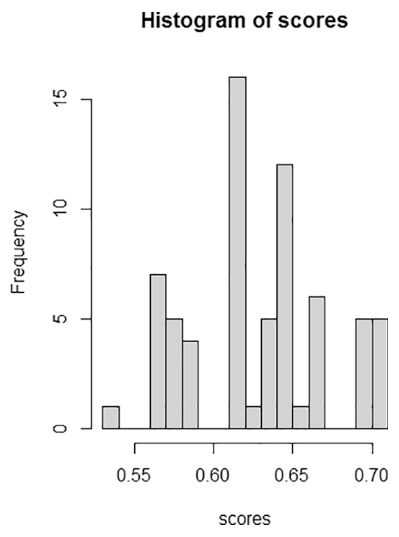

Fig. 12. Histogram of F1 scores on the validation set of individual Pima models.

Table 16
F1 scores on the validation set of PIMA data.

	center	corr	log	nzv	scale	zv
deepboost	0.6	0.6	0.6	0.6	0.6	0.6
LogitBoost	0.66	0.66	0.66	0.66	0.66	0.66
naive_bayes	0.59	0.59	0.64	0.59	0.59	0.59
rocc	0.6	0.6	0.59	0.6	0.58	0.6
xgbLinear	0.67	0.67	0	0.67	0.67	0.67
xgbTree	0.66	0.66	0	0.66	0.66	0.66

Table 17
Precision scores on the validation set of PIMA data.

	center	corr	log	nzv	scale	zv
deepboost	0.72	0.72	0.72	0.72	0.72	0.72
LogitBoost	0.71	0.71	0.71	0.71	0.71	0.71
naive_bayes	0.65	0.65	0.59	0.65	0.65	0.65
rocc	0.75	0.75	0.69	0.75	0.7	0.75
xgbLinear	0.73	0.73	0	0.73	0.73	0.73
xgbTree	0.79	0.79	0	0.79	0.79	0.79

Table 18
Recall scores on the validation set of PIMA data.

	center	corr	log	nzv	scale	zv
deepboost	0.52	0.52	0.52	0.52	0.52	0.52
LogitBoost	0.61	0.61	0.61	0.61	0.61	0.61
naive_bayes	0.54	0.54	0.7	0.54	0.54	0.54
rocc	0.51	0.51	0.52	0.51	0.49	0.51
xgbLinear	0.62	0.62	0	0.62	0.62	0.62
xgbTree	0.56	0.56	0	0.56	0.56	0.56

prudent to either change the validation data by selecting it from a basket of validation data or find new validation data at the absolute end of the pipeline. This potential flaw can also be countered by increasing the size of the data making this more unlikely. However, this is not always practically feasible.

RQ3 – To what extent can we automate the development of classification models?

The extent to which automation can be performed is contingent on the requirements and available resources. It is possible to automate a large pipeline with a small number of simple building blocks or to automate one step in this process heavily with the implementation of all possible approaches and niche situations. However, trying to combine this in one framework can drastically increase complexity if not handled intelligently. In the cases discussed in this study, however, it is shown that simple pipeline automation can be implemented to get comparable results as to what individual development can do. While this framework does not optimize as an individual could with enough time and understanding, it is a lot quicker and understandable for beginner data scientists as this framework repeats the standard workflow of data scientists to a certain degree. It allows for quick and simple model development providing clues to better manual improvement if desired.

The framework allows for rapid model creation with customization options but lacks the standard integration of more advanced techniques, especially with regard to feature engineering techniques such as Multi-relational decision tree learning or deep feature synthesis. It is, however, applicable for fast deployment with easy-to-understand and well-established techniques for a shotgun approach for finding better classification models.

The HRMS data set of urine profiles is extracted from the HRMS data in this paper in undisclosed ways. This data is derived from MS graphs resulting in intensities for certain chemical compositions per sample. The chemical compositions tested are containing exact masses corresponding to hormone masses. The HRMS data set of urine profiles are not publicly available due to confidentiality reasons and is used as-is in this study. The Pima data is publicly available and used as a reference to test the framework's capabilities.

6.2. *Threats to Validity*

This study has not conducted an exhaustive Systematic Literature Review. In addition to the implementation of the framework, care has been taken to ensure validity in terms of cross-referencing models and their corresponding preparations by using a well-established cross-validation technique. The framework is set up with a seed for every random function to ensure repeatability with the same data, packages, and environment. Widely used evaluation metrics and strategies have been applied, and therefore, the analysis of the results is reliable and repeatable.

Several feature engineering techniques, namely data imputation, transformation, feature selection, and data balancing are supported in the implemented framework. As known in the machine learning community, the effect of each feature engineering technique on the overall performance is highly sensitive to the nature of the application data at hand and there is no best way to be good in all of these fields. The proposed framework evaluates all possible combinations of the available algorithms for each technique automatically and later, reveals the optimal model considering the best score of the generated models. As there is a limited number of algorithms for each category in the current framework, the optimal model should be interpreted as the optimal within the framework context, which means that there might be additional algorithms, which were not been implemented in the framework yet, that can provide better overall performance. The framework is flexible to add new algorithms for each category of feature engineering techniques, therefore, there is no strict limitation on the number of algorithms used for each feature engineering technique.

6.3. *Comparison of the framework*

In this section, we compare our framework with a new machine learning framework called the Tidymodels framework (www.tidymodels.org), which is considered to be the successor of the Caret package. Both frameworks can be used by R programming users and have different features. In Table 19, we present our comparison using different aspects of these tools. Both frameworks were used by the researchers to implement different machine learning models and evaluations were reflected in this table. Since our framework was built on top of the Caret package, we benefit from the available features of the Caret

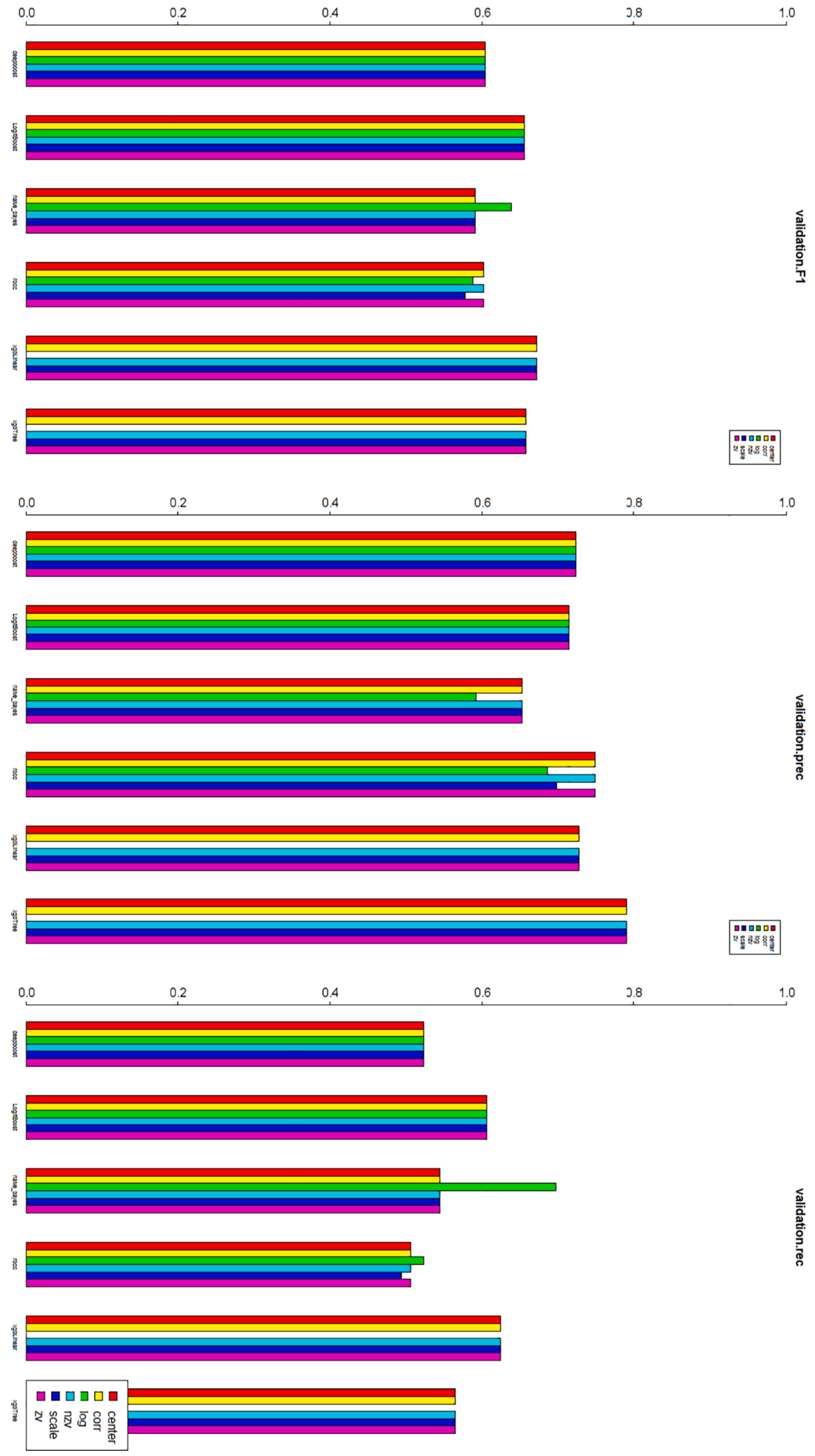

Fig. 13. Model performance metrics for the PIMA dataset averaged over two split ratios.

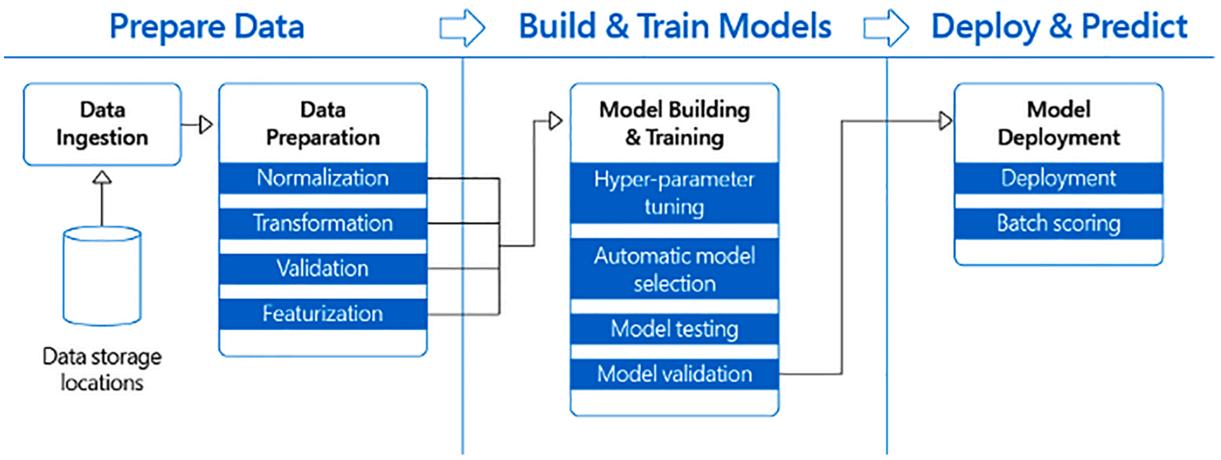

Fig. 14. Common data scientist workflow for making models (Lazzeri, 2022).

Table 19
Comparison of the framework with the state-of-the-art.

Evaluation Dimension	Proposed Framework	Tidymodels Framework
Learning Curve	Quick solutions	Steep learning curve
Stability	Stable	Still under development
Available Resources	More resources because of the CARET	Limited documentation
Automation of Model Selection	Semi-automated	Not automated
Complexity	Simple interfaces	Many steps and objects required
Implementation	Feels like standard R language	Pipe oriented
Available Algorithms	Many more algorithms	Collection of tools for machine learning

package such as available resources and the simplicity of the models.

7. Conclusion and future work

While alternatives are available for separate parts of the machine learning workflow, the proposed framework can help to find a highly accurate model in a short time without much expertise. For the imbalanced dataset, addressing this problem is an important factor in increasing model prediction qualities and should not be overlooked. Data transformation is not always beneficial but might provide a significant advantage depending on the underlying dataset. It is easier to create explainable Naïve Bayes or tree-based classifiers than deep learning-based models such as deepboost (Cortes et al., 2014). For a quick model generation, it is better to select algorithms that have simple asymptotic complexities. For example, the Naïve Bayes classifier is very fast in comparison to deepboost and the gradient boosting algorithms and it is able to reach similar performance if the right transformation step is chosen. This also indicates the importance of feature engineering and data preparation as it can greatly increase performance. This framework allows for quick exploration of these possibilities but is by no means exhaustive. It can be expanded upon by the use of recipes, that allow for greater automation. Therefore, the extent to which it can be automated is the extent to which methods can be implemented as recipes. More recipes allow for more comparisons. Combinations of recipes allow for greater chances of finding better-performing models.

Data Availability Statement

Data is available at the following webpage: https://github.com/mooym001/machine_learning. Due to confidentiality reasons, the sample names have been adapted.

CRediT authorship contribution statement

Sjoerd Boeschoten: Conceptualization, Methodology, Software, Validation, Writing – review & editing. **Cagatay Catal:** Conceptualization, Methodology, Validation, Writing – review & editing. **Bedir Tekinerdogan:** Conceptualization, Methodology, Validation, Writing – review & editing. **Arjen Lommen:** Conceptualization, Methodology, Validation, Writing – review & editing. **Marco Blokland:** Methodology, Validation, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data is available at the following webpage: https://github.com/mooym001/machine_learning. Due to confidentiality reasons, the sample names have been adapted.

Acknowledgment

Open Access funding provided by the Qatar National Library.

References

12 Using Recipes with train 12 Using Recipes with train | The caret Package. (n.d.). Retrieved January 7, 2022, from http://topepo.github.io/caret/using-recipes-with-train.html.
Atramentov, A., Leiva, H., & Honavar, V. (2003, September). In *A multi-relational decision tree learning algorithm–implementation and experiments* (pp. 38–56). Berlin, Heidelberg: Springer.
Azur, M. J., Stuart, E. A., Frangakis, C., & Leaf, P. J. (2011). Multiple imputation by chained equations: What is it and how does it work? *International Journal of Methods in Psychiatric Research, 20*(1), 40–49. https://doi.org/10.1002/MPR.329
Benedetto, A., Pezzolato, M., Robotti, E., Biasibetti, E., Poirier, A., Dervilly, G., … Bozzetta, E. (2021). Profiling of transcriptional biomarkers in FFPE liver samples: PLS-DA applications for detection of illicit administration of sex steroids and clenbuterol in veal calves. *Food Control, 128*, Article 108149.
Benedetto, A., Pezzolato, M., Biasibetti, E., & Bozzetta, E. (2021). Omics applications in the fight against abuse of anabolic substances in cattle: Challenges, perspectives and opportunities. *Current Opinion in Food Science, 40*, 112–120.
Bianco, M., Calvano, C. D., Ventura, G., Losito, I., & Cataldi, T. R. (2022). Determination of hidden milk allergens in meat-based foodstuffs by liquid chromatography coupled to electrospray ionization and high-resolution tandem mass spectrometry. *Food Control, 131*, Article 108443.
Bouwmeester, R., Gabriels, R., Van Den Bossche, T., Martens, L., & Degroeve, S. (2020). The Age of Data-Driven Proteomics: How Machine Learning Enables Novel Workflows. *Proteomics, 20*(21–22), 1–6. https://doi.org/10.1002/pmic.201900351

Breuker, K., Jin, M., Han, X., Jiang, H., & Mclafferty, F. W. (2008). FOCUS: DESORPTION IONIZATION AND MACROMOLECULAR MS Top-Down Identification and Characterization of Biomolecules by Mass Spectrometry. *J Am Soc Mass Spectrom, 19*, 1045–1053. https://doi.org/10.1016/j.jasms.2008.05.013

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357. https://doi.org/10.1613/JAIR.953

Chemyx. (2021). Basic Principles of HPLC, MS & LC-MS | Chemyx Inc. https://www.chemyx.com/support/knowledge-base/applications/basic-principles-hplc-ms-lc-ms/.

Cortes, C., Mohri, M., & Syed, U. (2014, June). Deep boosting. In International conference on machine learning (pp. 1179-1187). PMLR.

Crookston, N. L., & Finley, A. O. (2008). yaImpute: An R package for kNN imputation. *Journal of Statistical Software, 23*, 1–16.

Gibert, D., Planes, J., Mateu, C., & Le, Q. (2022). Fusing feature engineering and deep learning: A case study for malware classification. *Expert Systems with Applications, 207*, Article 117957.

Heaton, J. (n.d.). An Empirical Analysis of Feature Engineering for Predictive Modeling.

Hoffmann, W., Jackson, G. P., Hoffmann, W. D., & Jackson, G. P. (2008). Forensic Mass Spectrometry Clinical and Forensic Markers in Human Hair View project Fragmentation of oligosaccharides in tandem mass spectrometry View project Forensic Mass Spectrometry. *The Annual Review of Analytical Chemistry Is Online Annual Review of Analytical Chemistry, 8*, 419–440. https://doi.org/10.1146/annurev-anchem-071114-040335

Ilyas, I. F., & Rekatsinas, T. (2022). Machine Learning and Data Cleaning: Which Serves the Other? *ACM Journal of Data and Information Quality (JDIQ), 14*(3), 1–11.

It's All About the Features – Reality AI. (n.d.). Retrieved January 7, 2022, from https://reality.ai/its-all-about-the-features/.

Jollife, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065). https://doi.org/10.1098/RSTA.2015.0202.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science, 349*(6245), 255–260. https://doi.org/10.1126/SCIENCE.AAA8415

Liebal, U. W., Phan, A. N. T., Sudhakar, M., Raman, K., & Blank, L. M. (2020). Machine learning applications for mass spectrometry-based metabolomics. *Metabolites, 10*(6), 1–23. https://doi.org/10.3390/metabo10060243

Kanter, J. M., & Veeramachaneni, K. (2015). Deep feature synthesis: Towards automating data science endeavors. Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015. https://doi.org/10.1109/DSAA.2015.7344858.

Khatir, A. A. H. A., & Bee, M. (2022). Machine Learning Models and Data-Balancing Techniques for Credit Scoring: What Is the Best Combination? *Risks, 10*(9), 1–22.

Kuhn, M. (2008). Building Predictive Models in R Using the caret Package. *Journal of Statistical Software, 28*(5), 1–26. https://doi.org/10.18637/JSS.V028.I05

Lazzeri, F. (2022). *How to accelerate DevOps with Machine Learning lifecycle management*. Retrieved January 7, 2022, from https://web.archive.org/web/20200807120552/https://medium.com/microsoftazure/how-to-accelerate-devops-with-machine-learning-lifecycle-management-2ca4c86387a0.

Loos, G., Schepdael, A. Van, & Cabooter, D. (n.d.). Quantitative mass spectrometry methods for pharmaceutical analysis. https://doi.org/10.1098/rsta.2015.0366.

Lunardon, N., Menardi, G., & Torelli, N. (2014). ROSE: A Package for Binary Imbalanced Learning. *Undefined, 6*(1), 79–89. https://doi.org/10.32614/RJ-2014-008

Muhamedyev, R., Muhamedyev, R. I., Pushkina, K., & Kazakhstan, A. (2015). Machine learning methods. *An overview, 6*, 14–29. www.cmnt.lv.

Neves, D. T., Alves, J., Naik, M. G., Proença, A. J., & Prasser, F. (2022). From Missing Data Imputation to Data Generation. *Journal of Computational Science, 61*, Article 101640.

Pal, B., Tarafder, A. K., & Shahinur Rahman, M. D. (2020, January 10). Synthetic samples generation for imbalance class distribution with LSTM recurrent neural networks. PervasiveHealth: Pervasive Computing Technologies for Healthcare. https://doi.org/10.1145/3377049.3377115.

Petrovic, M., Díaz-Cruz, S., Barcelo, D., Petrovi, M., Dolores Hernando, M., Silvia Díaz-Cruz, M., & Barceló, D. (2005). Liquid Chromatography-Tandem Mass Spectrometry for the Analysis of Pharmaceutical Residues in Environmental Samples: A Review SedNet View project Globaqua View project Liquid chromatography-tandem mass spectrometry for the analysis of pharmaceutical residues in environmental samples: A review. *Article in Journal of Chromatography A, 1067*, 1–14. https://doi.org/10.1016/j.chroma.2004.10.110

Qaid, M. M., & Abdoun, K. A. (2022). Safety and concerns of hormonal application in farm animal production: A review. *Journal of Applied Animal Research, 50*(1), 426–439.

Rajawat, J., & Jhingan, G. (2019). Mass spectroscopy. *Data Processing Handbook for Complex Biological Data Sources, 1–20*. https://doi.org/10.1016/B978-0-12-816548-5.00001-0

Rinkal, P., & Aluvalu, R. (2014). A Multi-Relational Decision Tree Learning (MRDTL) Approach: A Survey. *IOSR Journal of Computer Engineering, 16*(2), 74–81. https://doi.org/10.9790/0661-16297481

Rocha, D. G., Lana, M. A. G., de Assis, D. C., de Macedo, A. N., Corrêa, J. M., Augusti, R., & Faria, A. F. (2022). A novel strategy for the detection of boldenone undecylenate misuse in cattle using ultra-high performance liquid chromatography coupled to high resolution orbitrap mass spectrometry: From non-targeted to targeted. *Drug Testing and Analysis, 14*(4), 667–675.

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., & Johannes, R. S. (1988, November). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the annual symposium on computer application in medical care (p. 261). American Medical Informatics Association.

Topçuoğlu, B. D., Lapp, Z., Sovacool, K. L., Snitkin, E., Wiens, J., & Schloss, P. D. (2021). mikropml: User-friendly R package for supervised machine learning pipelines. *Journal of open source software, 6*(61).

Vaishali Advani. (2021). What is Machine Learning | Definition, Tools, how it Works & Uses. https://www.mygreatlearning.com/blog/what-is-machine-learning/.

Van Buuren, S., & Oudshoorn, C. G. M. (2000). Multivariate Imputation by Chained Equations: MICE V1. 0 Users's Manual: TNO Prevention and Health. *Public Health.*

Yan, Y., & Maintainer,]. (2016). Package "MLmetrics, Machine Learning Evaluation Metrics.