This is a post-peer-review, pre-copyedit version of an article published in Machine Learning and Knowledge Extraction.
The final authenticated version is available online at:
https://doi.org/10.1007/978-3-030-57321-8_16

ML-ModelExplorer: An explorative model-agnostic approach to evaluate and compare multi-class classifiers

Andreas Theissler[1 [0000−0003−0746−0424]], Simon Vollert[1], Patrick Benz[1], Laurentius A. Meerhoff[2 [0000−0003−4386−0919]], and Marc Fernandes[1]

[1]Aalen University of Applied Sciences, 73430 Aalen, Germany
[2] Leiden Institute of Advanced Computer Sciences (LIACS), Leiden University, Leiden, The Netherlands.

Abstract. A major challenge during the development of Machine Learning systems is the large number of models resulting from testing different model types, parameters, or feature subsets. The common approach of selecting the best model using one overall metric does not necessarily find the most suitable model for a given application, since it ignores the different effects of class confusions. Expert knowledge is key to evaluate, understand and compare model candidates and hence to control the training process. This paper addresses the research question of how we can support experts in the evaluation and selection of Machine Learning models, alongside the reasoning about them. ML-ModelExplorer is proposed – an explorative, interactive, and model-agnostic approach utilising confusion matrices. It enables Machine Learning and domain experts to conduct a thorough and efficient evaluation of multiple models by taking overall metrics, per-class errors, and individual class confusions into account. The approach is evaluated in a user-study and a real-world case study from football (soccer) data analytics is presented.
ML-ModelExplorer and a tutorial video are available online for use with own data sets: `www.ml-and-vis.org/mex`

Keywords: multi-class classification · model selection · feature selection · human-centered Machine Learning · visual analytics

1 Introduction

During the development of Machine Learning systems a large number of model candidates are generated in the training process [27] by testing different model types, hyperparameters, or feature subsets. The increased use of Deep Learning [19] further aggravates this problem, as the number of model candidates is very large due to the enormous number of parameters. This paper is motivated by the following observations for multi-class classification problems:

1. Automatically selecting the best model based on a single metric does not necessarily find the model that is best for a specific application, e.g. different per-class errors have different effects in applications.

2. In applications with uncertain relevance and discriminative power of the given input features, models can be trained on different feature subsets. The results of the models are valuable for drawing conclusions about the feature subsets, since the level of contribution of a feature to the classification performance is unknown a priori.

We argue that expert knowledge is key to evaluate, understand and compare model candidates and hence to control the training process. This yields the research question: *How can we support experts to efficiently evaluate, select and reason about multi-class classifiers?*

This paper proposes ML-ModelExplorer which is an explorative and interactive approach to evaluate and compare multi-class classifiers, i.e. models assigning data points to $N > 2$ classes. ML-ModelExplorer is model-agnostic, i.e. works for any type of classifier and does not evaluate the inner workings of the models. It solely uses the models' confusion matrices and enables the user to investigate and understand the models' results. It allows to take different overall metrics, the per-class errors, and the class confusions into account and thereby enables a thorough and efficient evaluation of models.

We believe that in a multi-class problem – due to the high number of errors per model, per class and class confusions – interactively analysing the results at different levels of granularity, is more efficient and will yield more insights than working with raw data. This hypothesis is evaluated with a user study and a real-world case study. In the case study on football data, we examine how successful attacking sequences from different parts of the pitch can best be modelled using a broad range of novel football-related metrics.

This paper makes the following contributions:

1. Provision of a brief review of the state-of-the-art in visual analysis of multi-class classifier results (Section 2).
2. Proposal of a data- and model-agnostic approach for the evaluation, comparison and selection of multi-class classifiers (Section 3, Section 4).
3. Evaluation with a user study (Section 5).
4. Validation of real-world relevance with a case study (Section 6).
5. Supply of ML-ModelExplorer for use with own data sets.

2 Related work

The general interplay of domain experts and Machine Learning has been suggested in literature, for example in [14,10,3,30]. Closely related to the problem discussed in this paper, approaches to interactively analyse the results in multi-class problems have been proposed and are briefly surveyed in the following.

In [29] the authors proposed Squares which integrates the entire model evaluation workflow into one visualisation. The core element is a sortable parallel coordinates plot with the per-class metrics, enhanced by boxes showing the classification results of instances and the thumbnails of the images themselves. ConfusionWheel [1] is a radial plot with classes arranged on an outer circle,

class confusions shown by chords connecting the classes, and histograms on the circle for the classification results of all classes. ComDia+ [23] uses the models' metrics to rank multiple image classifiers. The visualization is subdivided into a performance ranking view resembling parallel coordinates, a diagnosis matrix view showing averaged misclassified images, and a view with information about misclassified instances and the images themselves. Manifold [33] contrasts multiple models, showing the models' complementarity and diversity. Therefor it uses scatter plots to compare models in a pairwise manner. A further view indicates the differences in the distributions of the models at the feature level. INFUSE [16] is a dashboard for the selection of the most discriminative features in a high-dimensional feature space. The different feature rankings across several feature ranking methods can be interactively compared.

The idea from [1] of using a radial plot was used in one visualisation in ML-ModelExplorer. While Squares [29] and ConfusionWheel [1] are designed for the evaluation of a single classifier, ML-ModelExplorer has the focus of contrasting multiple models. ComDia+ [23] and Manifold [33] allow to compare multiple models, where the view of averaged images constrains ComDia+ to (aligned) images. Manifold is a generic approach which in addition to classification is applicable to regression. While the mentioned approaches allow to refine the analyses to instance or feature level, they require the input data. ML-ModelExplorer solely works on the models' confusion matrices and is hence applicable to the entire range of classification problems, not constrained to data types like images or feature vectors. A further reason not to incorporate the data set itself is, that having to upload their data into an online tool will discourage practitioners and researchers from testing the approach. In addition, in contrast to some of the aforementioned approaches, ML-ModelExplorer is made publicly available.

3 Problem analysis: Evaluating multi-class classifiers

For a multi-class classification problem the approach incorporates all tested models M into the user-driven analysis. The output of a multi-class classification problem with $|C|$ classes denoted as C_i can be presented as a confusion matrix of dimension $|C| \times |C|$ (see e.g. [15]). The $|C|$ elements on the diagonal show the correct classifications, the remaining elements show the $|C| \times (|C| - 1)$ different confusions between classes. An example is shown in Table 1, where in this paper columns correspond to class labels and rows show the predictions.

	class label C_1	class label C_2	class label C_3
prediction C_1	70 (0.7)	30 (0.15)	0 (0.0)
prediction C_2	20 (0.2)	150 (0.75)	50 (0.1)
prediction C_3	10 (0.1)	20 (0.1)	450 (0.9)

Table 1. A confusion matrix of an imbalanced three-class classification problem with absolute numbers and percentages (0...1).

From the confusion matrix, a variety of metrics can be deduced [15,6]. In the following, the metrics relevant for this paper are introduced. The absolute number of correctly classified instances of class C_i is referred to as true predictions and denoted as TP_{C_i}. The true prediction rate – also termed recall or true positive rate – for class C_i is denoted as TPR_{C_i} and is the percentage of correctly classified instances of class C_i:

$$TPR_{C_i} = \frac{TP_{C_i}}{|C_i|} \tag{1}$$

The per-class error E_{C_i} is given by:

$$E_{C_i} = 1 - TPR_{C_i} \tag{2}$$

The overall accuracy acc refers to the percentage of correctly classified instances of all classes C_i

$$acc = \frac{1}{N} \sum_{i=1}^{|C|} TP_{C_i} \tag{3}$$

Taking into account potential class imbalance in the data set, can be achieved with the macro-average recall $recall_{avg}$ which is the average percentage of correctly classified instances of all classes C_i:

$$recall_{avg} = \frac{1}{|C|} \sum_{i=1}^{|C|} TPR_{C_i} \tag{4}$$

The common approach of model selection based on a single metric, e.g. overall or weighted accuracy, does not necessarily find the most suitable model for an application, where different class confusions have different effects [11]. For example a model might be discarded due to a low accuracy caused by frequently confusing just two of the classes, while being accurate in detecting the remaining ones. This model can be of use for (a) building an ensemble [25,32], (b) by refining it with regard to the two confused classes, or (c) to uncover mislabelling in the two classes.

Target users of ML-ModelExplorer are (1) *Machine Learning experts* for evaluation, refinement, and selection of model candidates, and (2) *domain experts*[1] for the selection of appropriate models for the underlying application and for reasoning about the discriminative power of feature subsets.

Based on the authors' background in Machine Learning projects and the discussion with further experts, Scrum user stories were formulated. These user stories guided the design of ML-ModelExplorer.

- **User story #1 (Overview)**: As a user I want an overview that contrasts the results of all $|M|$ models, so that I can find generally strong or weak models as well as similar and outlier models.

[1] domain experts are assumed to have a basic understanding of classification problems, i.e. understand class errors and class confusions

- **User story #2 (Model and class query)**: As a user I want to query for models based on their per-class errors, so that I can understand which classes lead to high error rates over all $|M|$ models and which classes cause problems only to individual models.
- **User story #3 (Model drill-down)**: As a user I want to drill-down into the details of one model M_k, so that I can conduct a more detailed analysis e.g. regarding individual class confusions.
- **User story #4 (Model comparison)**: As a user I want to be able to select one model M_k and make a detailed comparison with a reference model M_r or the average of all models M_{avg}, so that I can understand where model M_k needs optimization.

4 The approach: ML-ModelExplorer

ML-ModelExplorer provides an overview of all models and enables interactive detailed analyses and comparisons. Starting with contrasting different models based on their overall metrics, more detailed analyses can be conducted by goal-oriented queries for models' per-class results. A model's detailed results can be investigated and compared to selected models.

ML-ModelExplorer uses a variety of interactive visualisations with highlighting, filtering, zooming and comparison facilities that enable users to (1) select the most appropriate model for a given application, (2) control the training process in a goal-oriented way by focusing on promising models and further refining them, and (3) reason about the effect of features, in the case where the models were trained on different feature subsets.

The design was governed by the goal to provide an approach that does not require specific knowledge in data visualisation or the familiarisation with new visualisation approaches, since domain experts do not necessarily have a Data Science background. Consequently a combination of well-known visualisations, that can be reasonably assumed to be known by the target users, are used as key elements. Examples are easily interpretable scatter plots, box plots, bar charts, tree maps, and chord diagrams. In addition some more advanced – but common – interactive visualisations were utilised, i.e. parallel coordinates or the hierarchical sun burst diagram. In order to compensate differing prior knowledge or preferences of users, the same information is redundantly communicated with different visualisations, i.e. there are multiple options to conduct an analysis.

ML-ModelExplorer is implemented in R [26] with shiny [4] and plotly [12] and can be used online[2] with own data sets and a tutorial video[3] is available. Different characteristics of the models' results are emphasized with complementary views. The design follows Shneiderman's information seeking mantra [31], where *overview first* is achieved by a model overview pane. In order to incrementally refine the analysis, *zoom + filter* is implemented by a filtering facility for models and classes and by filtering and zooming throughout the different visualisations.

[2] ML-ModelExplorer online: `www.ml-and-vis.org/mex`
[3] ML-ModelExplorer video: `https://youtu.be/IO7IWTUxK_Y`

Details-on-demand is implemented by views on different detail levels, e.g. model details or the comparison of models. The design was guided by the user stories in Section 3, the mapping of these user stories to visualisations is given in Table 2.

User story	**Implemented by**
User story #1 (Overview)	per-model metrics plot model similarity plot
User story #2 (Model and class query)	per-class errors query view class error radar chart
User story #3 (Model drill-down)	error hierarchy plot confusion matrix view confusion circle bilateral confusion plot confusion tree map
User story #4 (Model comparison)	delta confusion matrix delta radar chart

Table 2. Mapping of user stories to interactive visualisations.

In the following, the views are introduced where screenshots illustrate an experiment with 10 convolutional neural networks (CNN) [28] on the MNIST data set [18], where the task is to classify the handwritten digits 0...9. CNNs with a convolution layer with 32 filters, a 2×2 max pooling layer, and a dropout of 0.2 were trained. The hyperparameters kernel size k, which specifies the size of the filter moved over the image, together with the stride were varied from $k = 2$ to $k = 11$, resulting in 10 model candidates denoted by $\text{M1\_CNN}_{k=2}$...$\text{M10\_CNN}_{k=11}$.

4.1 Model overview pane

The *model overview pane* (Fig. 1) is subdivided into three horizontal subpanes starting with (1) a coarse overview on the top showing the summarised metrics of all models, e.g. the overall accuracy and the dispersion of the true prediction rates TPR_{C_i} over all classes C_i. The middle subpane (2) contrasts and allows to query the per-class errors E_{C_i}. On the bottom (3), a detailed insight can be interactively gained by browsing the models' class confusions:

Model metrics subpane (1):

- *Per-model metrics plots:* This set of plots gives an overview of generally good or weak models (Fig. 1, 1), hence serving as a starting point to detect potential model candidates for further refinement or for the exclusion from the training process. The per-model metrics can be viewed at different levels of granularity with the following subplots: a list of ranked and grouped models,

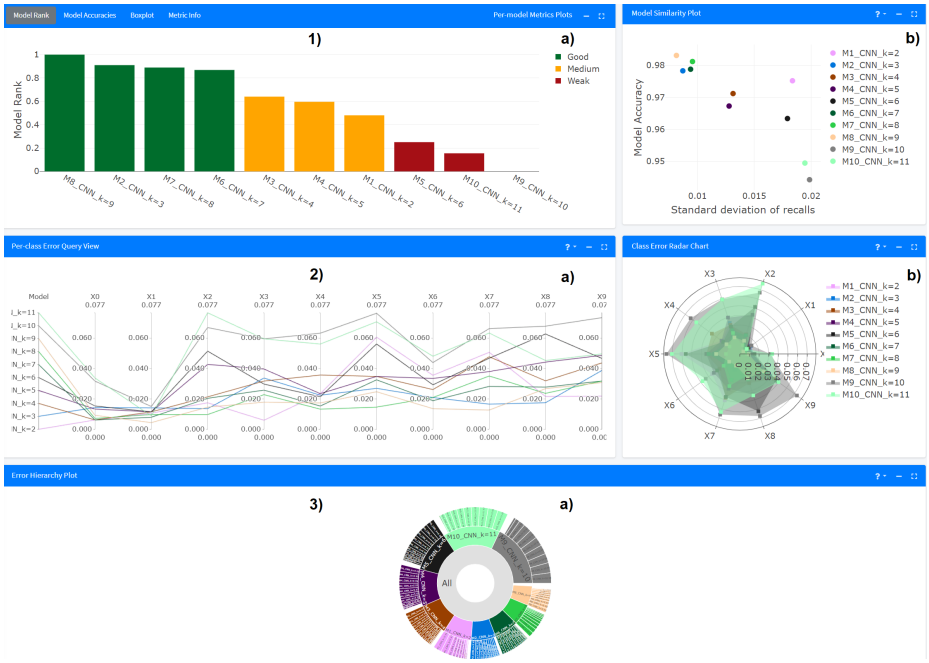

Fig. 1. Model overview pane contrasting the results of ten multi-class classifiers on the MNIST data set. The models are ranked according to their accuracy and classification imbalance in 1) a). The non-monotonic effect of the varied kernel size in the used convolutional neural networks can be seen in 2) a).

a line plot with the model accuracies, and a box plot with the dispersions of recall, precision and F1-score.

In the model rank subplot (Fig. 1, 1 a), the models are ranked and grouped into strong, medium and weak models. For the ranking of the models, the underlying assumption is, that a good model has a high accuracy and a low classification imbalance, i.e. all classes have a similarly high detection rate.

In the following, a metric to rank the models is proposed. In a first step, the classification imbalance CI is defined as eq. (5), which is the mean deviation of the true prediction rates TPR_{C_i} from the model's macro-average recall, where $CI = 0$ if TPR_{C_i} is identical for all C_i:

$$CI = \frac{1}{|C|} \sum_{i=1}^{|C|} |\, TPR_{C_i} - recall_{avg}| \tag{5}$$

Each model M_k is described by the vector $\gamma_k = (recall_{avg}, CI)$. To ensure equal influence of both metrics, the components of γ_k are min-max scaled over all models, i.e. $\gamma'_k = (recall'_{avg}, CI')$ with $recall'_{avg} = [0, 1]$ and $CI' = [0, 1]$. To assign greater values to more balanced models, CI' is substituted

by $1 - CI'$ and the L1-norm is then used to aggregate the individual metrics into a model rank metric:

$$M_{rank_k} = \frac{1}{2}|recall'_{avg} + (1 - CI')| \tag{6}$$

where $\frac{1}{2}$ scales M_{rank_k} to a range of $[0, 1]$, allowing for direct interpretation of M_{rank_k}. The models are ranked by M_{rank_k}, where $M_{rank_k} \to 1$ correspond to stronger models.

In addition to ranking, the models are grouped into $good, medium, weak$, which is beneficial for a high number of models. Since the categorization of good and weak models is not absolute, but rather dependent on the complexity of the problem, i.e. the data set, the grouping is conducted using reference models from M. From the $|M|$ models, the best model M_b and the weakest model M_w are selected as reference models utilizing M_{rank_k}. In addition a medium model M_{med} is selected, which is the median of the ranked models. Following that, the $|M|$ models are classified as any of $good, medium, weak$ using a 1-nearest neighbour classifier on M_{rank_k}, encoded with green, yellow, and red (see Fig. 1, 1 a)).

In the second subplot the models' accuracies are contrasted with each other in order to a allow a coarse comparison of the models. In addition the accuracies are shown in reference to a virtual random classifier, randomly classifying each instance with equal probability for each class, and to a baseline classifier, assigning all instances to the majority class.

In the third subplot, the models are contrasted using box plots showing the dispersions of the per-class metrics. Box plots positioned at the top indicate stronger models while the height indicates a model's variability over the classes. For the MNIST experiment, the strong and non-monotonic effect of the kernel size is visible, with 7, 8, and 9 yielding the best results and a kernel size of 6 having a high variability in the classes' precisions.

– *Model similarity plot:* This plot shows the similarities between models (Fig. 1, 1b). For each model the overall accuracy and the standard deviation of the true prediction rates $TPR_{C_i} \forall C_i$ are extracted, and presented in a 2D-scatter plot. Similar models are thereby placed close to each other, where multiple similar models might reveal clusters. Weak, strong, or models with highly different results become obvious as outliers. For the MNIST data set, the plot reveals a group of strong models, with high accuracies and low variability over the classes: M6_CNN$_{k=7}$, M7_CNN$_{k=8}$, M8_CNN$_{k=9}$, and M2_CNN$_{k=3}$.

Per-class errors subpane (2):

– *Per-class errors query view:* This view shows the errors E_{C_i} and allows to query for models and classes (Fig. 1, 2a). The errors are mapped to parallel coordinates [13] which show multi-dimensional relations with parallel axes and allow to highlight value ranges. The first axis shows the models, each class is mapped to one axis with low E_{C_i} at the bottom. For each model, line

segments connect the errors. For the MNIST data, the digits 0 and 1 have the lowest errors, while some of the models have high errors on 2 and 5..

– *Class error radar chart:* The per-class errors E_{C_i} can be interactively analysed in a radar chart, where the errors are mapped to axes (Fig. 1, 2b). The models' results can be rapidly contrasted, larger areas showing high E_{C_i}, and the shape indicating high or low errors on specific classes. The analysis can be incrementally refined by deselecting models. In the MNIST experiment, models with general weak performance are visible by large areas and it reveals that all models have high detection rates TPR_{C_i} on digit 0 and 1.

Class confusions subpane (3):

– *Error hierarchy plot:* This plot allows to navigate through all errors per model and class in one view (Fig. 1, 3a). The hierarchy of the overall errors for each model ($1 - recall_{avg}$), the per-class errors E_{C_i}, and the class confusions are accessible in a sun burst diagram. The errors at each level are ordered clockwise allowing to see the ranking. One finding in the MNIST experiment is, that the model with the highest accuracy (M8_CNN$_{k=9}$) has its most class confusions on digit 9, which is most often misclassified with 7, 4, and 1.

4.2 Model details pane

The *model details pane* shows different aspects of one selected model, here M3_CNN$_{k=4}$ (see Fig. 2). The following four plots are contained:

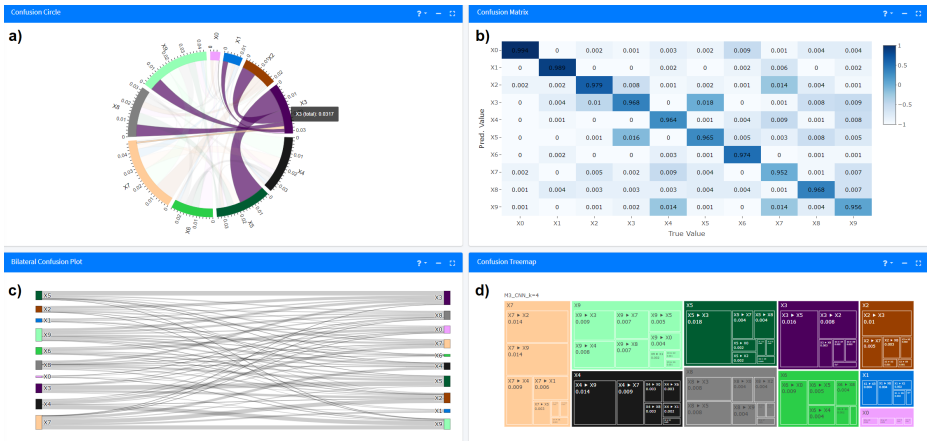

Fig. 2. Model details pane showing one selected model. Per-class errors and class confusions can be investigated.

– *Confusion circle:* This plot (Fig. 2, a) is a reduced version of the confusion wheel proposed in [1]. The $|C|$ classes are depicted by circle segments in one surrounding circle. The class confusions are shown with chords connecting the circle segments. The chords' widths encode the error between the classes. Individual classes can be highlighted and the detailed errors are shown on demand. For the MNIST experiment, the class confusions of M3_CNN$_{k=4}$ reveal that e.g. digit 3 is most often misclassified as 5, 8, and 9. On the other hand, of all digits misclassified as 3, digit 5 is the most frequent one.
– *Confusion matrix:* The model's confusion matrix is shown in the familiar tabular way, with a colour gradient encoding the class confusions (Fig. 2, b).
– *Bilateral confusion plot:* In an interactive Sankey diagram (Fig. 2), misclassifications can be studied (class labels on the left, predictions on the right), c). By rearranging and highlighting, the focus can be put on individual classes.
– *Confusion tree map:* A model's per-class errors E_{C_i} are ranked in a tree map allowing to investigate how E_{C_i} is composed of the individual class confusions, where larger areas correspond to higher errors (Fig. 2, d). If class C_i is selected, the ranked misclassifications to $C_1 ... C_{|C|}$ are shown. In the MNIST experiment, for M3_CNN$_{k=4}$ the weakness is digit 7, which in turn is most frequently misclassified as 9 and 2.

4.3 Model comparison pane

In the *model comparison pane* a model M_k can be selected and compared to a selected reference model M_r or to the average over all models M_{avg} (Fig. 3).

Fig. 3. Model comparison pane: A selected model can be compared with a selected reference model and with the average of all models.

– *Delta confusion matrix:* The class confusions of M_k can be contrasted to a reference model M_r showing where M_k is superior and where it needs optimization (Fig. 3, a). The difference between the class confusions is visible per cell with shades of green encoding where M_k is superior to M_r and red where M_r is superior, respectively. In the MNIST experiment, while M5_CNN$_{k=6}$ is in general the weaker model compared to M1_CNN$_{k=2}$, it less frequently misclassifies e.g. 7 as 9 and 5 as 8.

– *Delta error radar chart:* The differences in the per-class errors of a model M_k w.r.t. a reference model M_r and the average M_{avg} is illustrated in a radar chart (Fig. 3, b). The area and shape in the radar chart allows to rapidly draw conclusions about weak or strong accuracies on certain classes and about differences between the models. While in the MNIST experiment M1_CNN$_{k=2}$ has slightly higher errors on digits 5, 6, 7 and approximately similar errors on 0 and 1, it performs significantly better than M5_CNN$_{k=6}$ and the overall average on the remaining digits.

5 Evaluation: user study

A user study was conducted in order to compare ML-ModelExplorer to the common approach of working on a Machine Learning library's raw output. Python's scikit-learn [24] was used as a reference. Two typical activities were tested:

1. evaluation and selection of a single model
2. controlling the training process by identifying weak models to be discarded, strong models to be optimised, and by uncovering optimisation potential

For these two activities, hypotheses H1 and H2 were formulated and concrete typical tasks were defined in Table 3. The tasks were solved by two disjoint groups of users using either Python (group A) or ML-ModelExplorer (group B). While using two disjoint groups halves the sample size, it avoids a learning effect which is to be expected in such a setting. The efficiency is measured by the number of correct solutions found in a given time span, i.e. [correct solutions/minute].

Eighteen students with industrial background, enrolled in an extra occupational master course, participated in the user study. The participants had just finished a machine learning project with multi-class classification and had no specific knowledge about data visualisation. The 18 participants were randomly assigned to group A and B resulting in a sample size of $N = 9$ for a paired test. Group A was given a jupyter-notebook with the raw output pre-loaded into Python's pandas data structures and additionally text file with all confusion matrices and metrics. These participants were allowed to use the internet and a pandas cheat sheet was handed out. Group B used ML-ModelExplorer and a one-page documentation was handed out.

As a basis for the user study, the results of 10 different models on the MNIST data set [18] were used, as shown in Section 4. Prior to the study, the data, the supplied Python code and ML-ModelExplorer were briefly explained. A maximum of 25 minutes for the tasks of H1 and 15 minutes for H2 was set. The tasks were independently solved by the test participants, without intervention. No questions were allowed during the user study. The individual performances of the students are shown in Fig. 4. Note that the maximum possible score for the tasks within H1 was 3, whereas the maximum possible score for the tasks within H2 was 16. Therefore the efficiency of the students is calculated using the number of correct answers as well as the required time to solve the tasks.

$\mathbf{H1_{Null}}$	With ML-ModelExplorer the selection of the best model is *not* more efficient than with a ML library's raw output.
$\mathbf{H1_{Alternative}}$	With ML-ModelExplorer ... is more efficient.
$T1_1$	Find the model with the highest overall accuracy.
$T1_2$	Find the model with the lowest error on class 8.
$T1_3$	For model M8 find the two classes that class 8 is most frequently misclassified as.
$\mathbf{H2_{Null}}$	With ML-ModelExplorer controlling the training process is *not* more efficient than with a ML library's raw output.
$\mathbf{H2_{Alternative}}$	With ML-ModelExplorer ... is more efficient.
$T2_1$	Find the 3 models with the highest and the 3 with the lowest accuracies.
$T2_2$	For model M8, find the 3 classes with the highest per-class errors.
$T2_3$	For model M8, find the 3 pairs of classes most frequently confused.
$T2_4$	Find the classes, where M8 has a higher error than M6.
$T2_5$	Compare M8 and M6 and from the class confusions where M8 has a higher error, find the 2 with the highest differences.

Table 3. Hypotheses and typical tasks to be solved in the user study.

The distribution of the efficiencies is shown in Fig. 5, indicating that participants using ML-ModelExplorer were more efficient. In addition there appears to be a learning effect: for the tasks connected with hypothesis H2 the participants of both groups were more efficient than for H1.

The hypotheses $H1_{Null}$ and $H2_{Null}$ state that there is no statistically significant difference in the efficiencies' mean values between group A (raw output and python code) and group B (ML-ModelExplorer). One-sided paired t-tests were conducted with a significance level of $\alpha = 0.05$. The resulting critical values are $c_{H1} = 0.191$ and $c_{H2} = 0.366$. The observed differences in the user study are $\overline{x}_{B_{H1}} - \overline{x}_{A_{H1}} = 0.376$ and $\overline{x}_{B_{H2}} - \overline{x}_{A_{H2}} = 0.521$, where all values are given as [correct solutions/minute].

Hence, due to $(\overline{x}_{B_{H1}} - \overline{x}_{A_{H1}}) > c_{H1}$ and $(\overline{x}_{B_{H2}} - \overline{x}_{A_{H2}}) > c_{H2}$, both null hypotheses $H1_{Null}$ and $H2_{Null}$ were rejected, i.e. ML-ModelExplorer was found to be more efficient for both of the typical activities (1) evaluation and selection of a single best model and (2) controlling of the training process.

6 Case Study: Analysing tactics in football

In the following case study, the applicability of ML-ModelExplorer to real-world problems is evaluated with a multi-class classification problem on tracking data from football (soccer). In football, a recent revolution has been unchained with the introduction of position tracking data [22]. With the positions of all the players and the ball, it is possible to quantify tactics using the players' locations over time [21]. Machine Learning techniques can be adopted to fully exploit the opportunities tracking data provides to analyse tactical behaviour [9]. Although without a doubt Machine Learning will be a useful addition to the tactical anal-

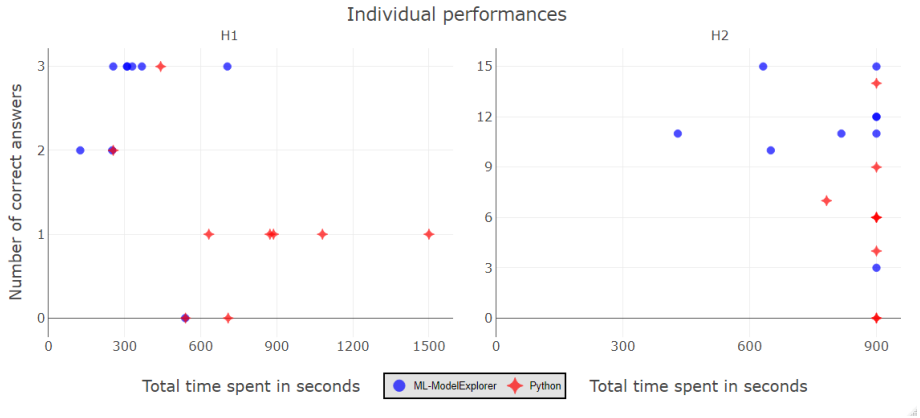

Fig. 4. The distribution of individual performances across both hypotheses.

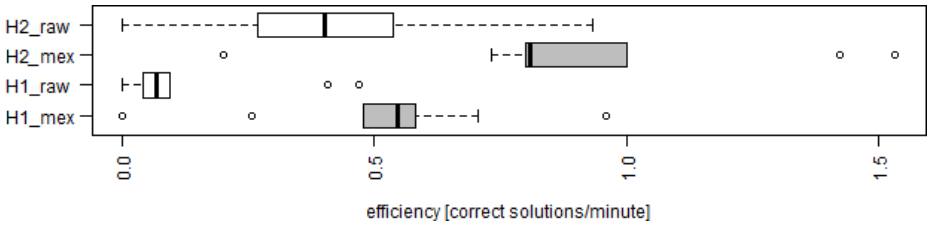

Fig. 5. The distribution of [correct solutions/minute] when analysing raw output with Python (H1_raw, H2_raw) and with ML-ModelExplorer (H1_mex, H2_mex).

yses, one of the major challenges is to involve the domain experts (i.e., the coaching staff) in the decision-making process. Engaging the domain expert in the model selection process is crucial for (fine-) tuning the models. Specifically, the domain expert can play an important role in identifying the least disruptive class confusions. With better models, and models that are more supported by the domain experts, Machine Learning for analysing tactics in football will be more quickly embraced.

Football is an invasion-based team sport where goals are rare events, typically 2-4 goals out of the 150-200 offensive sequences in a 90 minutes match [2]. It is thus important to find the right balance between creating a goal-scoring opportunity without weakening the defence (and giving the opponents a goal-scoring opportunity). For example, a team could adopt a compact defence (making it very difficult for the opponents to score, even if they outclass the defending team) and wait for the opponent to lose the ball to start a quick counter-attack. If the defence is really compact, the ball is usually recovered far away from the attacking goal, whereas a team that puts a lot of pressure across the whole pitch might recover the ball in a promising position close to the opponent's goal. To

formulate an effective tactic, analysts want to know the consequences of losing (or regaining) the ball in specific parts of the pitch. This raises the question: How can successful and unsuccessful attacks that started in different parts of the pitch (see Fig. 6) be modelled?

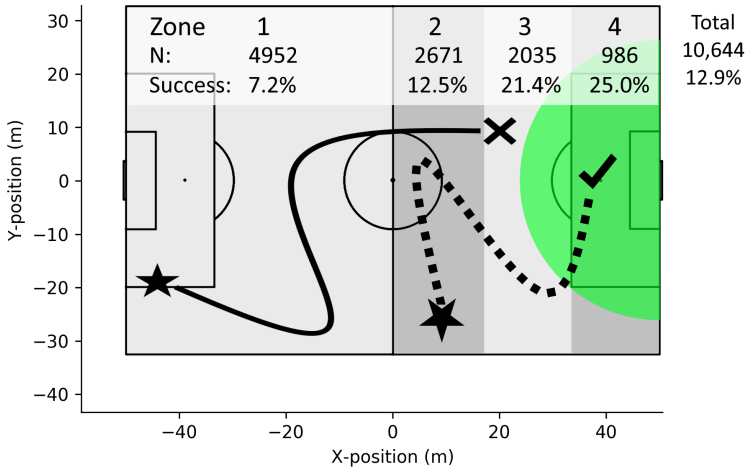

Fig. 6. Visualisation of the 8 classes based on a combination of zone (1-4) and attack outcome (success, fail, that is in-/outside green area, respectively). Example trajectories of the ball demonstrate an unsuccessful attack from zone 1 ($Z1_{fail}$, solid line) and a successful attack from zone 2 ($Z2_{success}$, dotted line).

6.1 Procedure

The raw data contain the coordinates of each player and the ball recorded at 10 Hz with an optical tracking system (SportsVU, STATS LLC, Chicago, IL, USA). We analysed 73 matches from the seasons 2014-2018 from two top-level football clubs in the Dutch premier division ('Eredivisie'). For the current study, we analysed attacking sequences, which were defined based on team ball possession. Each attacking sequence was classed based on *where an attack started* and *whether it was successful*. The starting locations were binned into 4 different zones (see the 'stars' in Fig. 6). To deal with the low number of truly successful attacks (i.e., goals scored), we classed each event using the distance to the goal at the end of an attack as a proxy for success: Successful attacks ended within 26 m to the centre of the goal (see the green shaded area in Fig. 6).

For each event, we computed 72 different metrics that capture football tactics [21]. Some of these metrics describe the spatial distribution of the players on the pitch [7]. Other metrics capture the disruption of the opposing team (i.e., how much they moved in response to an action of the other team) [8]. Lastly, we created a set of metrics related to the ball carrier's danger on the pitch (i.e.,

"Dangerousity" [20]), which is a combination of four components. *Control* measures how much control the player has over the ball when in possession. *Density* quantifies how crowded it is between the ball carrier and the goal. *Pressure* captures how closely the ball carrier is defended. Lastly, *Zone*, refers to where on the pitch the ball carrier is, where players closer to the goal get a higher value than players further away. Note that the metric *Zone* only has values for the last part of the pitch, which corresponds to zone 4 of zones used for the target.

Finally, we combined the classed events (i.e., attacking sequences) with the tactical metrics by aggregating the temporal dimension by, for example, averaging across various windows prior to the end of the event. The resulting feature vectors were grouped based on whether the metrics described the *Spatial* distribution of the players on the pitch (n = 1092), *Disruption* (n = 32), *Control* (n = 46), *Density* (n = 46), *Pressure* (n = 46), *Zone* (n = 46), *Dangerousity* (n = 46), and all *Link*'s Dangerousity-related metrics combined (n = 230). Subsequently, we trained five different classifiers (decision tree, linear SVC, k-NN, extra trees, and random forest) with each of the eight feature vectors, yielding 40 different models to evaluate with ML-ModelExplorer.

6.2 Model Evaluation with ML-ModelExplorer

An exploration of all models reveals the large variation in accuracy (see Fig. 7). The difficulty of modelling tactics in football is apparent given how many of the models are under Baseline Accuracy (i.e., majority class). Particularly the Random Forests do well in this Machine Learning task. Next to the modelling techniques, the different features subsets also yield varying model accuracies. The Random Forests with *Density*, *Zone*, *Dangerousity*, *Link* and *Spatial* clearly outperform the other subsets.

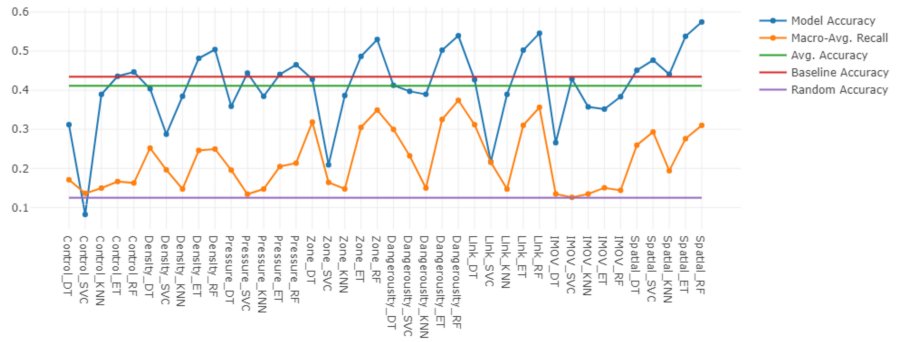

Fig. 7. An overview of the quality of all 40 models where an accuracy above baseline indicates that the model performed better than simply taking the majority class.

By selecting the models with the highest accuracies, the per-class errors can be easily compared (see Fig. 8). Two of the models, *Spatial RF* and *Density RF*

have more class confusions with the successful attacking sequences. Here, the class-imbalance seems to play a role: the more interesting successful attacking sequences occur much less frequently than the unsuccessful attacking sequences. This becomes even more evident when using the built-in function to switch to relative numbers (see Fig. 8, left and right panel, respectively).

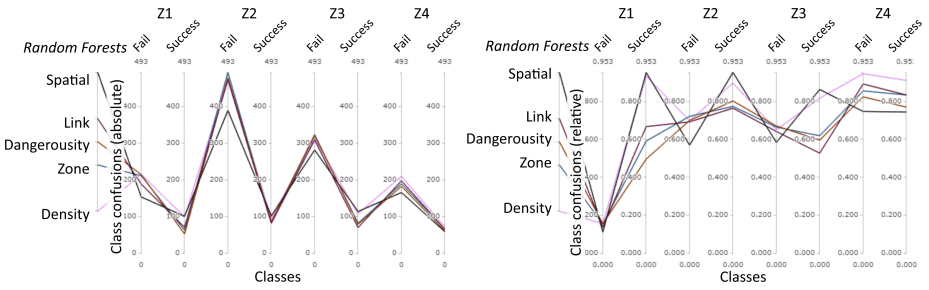

Fig. 8. Absolute (left) and relative (right) class confusions of the most promising models.

In fact, a domain expert might put more or less importance to specific classes. In this case, a football analyst would not be interested in unsuccessful attacks starting in zone 1 (which also happens to be the bulk of the data). Using another of the ML-Explorer's built-in functions, the domain expert can deselect 'irrelevant' classes. By excluding the unsuccessful attacks from zone 1 (i.e., "$Z1_{fail}$"), a clear difference in the best performing models becomes apparent: the *Spatial*-related models perform worse than average when the least interesting class is excluded (see Fig. 9). For a domain expert this would be a decisive difference to give preference to a model that might not have the highest overall accuracy.

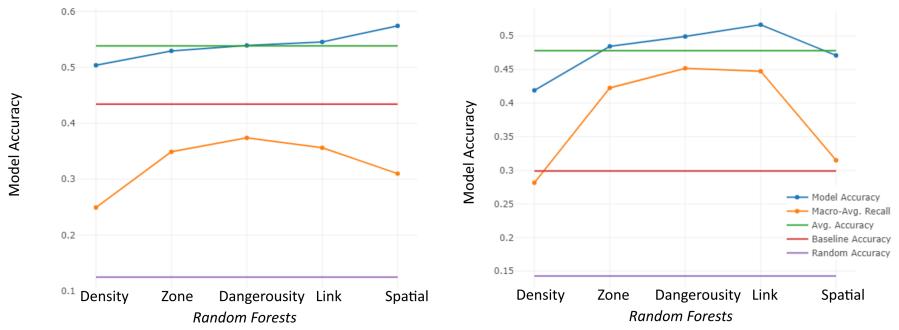

Fig. 9. The model qualities for the 5 most promising models with all classes included (left) and the less relevant class $Z1_{fail}$ excluded (right).

By now, it is clear that the models have different strengths and weaknesses. Some of the models perform well on the unsuccessful attacking sequences starting further away from the opponent's goal (i.e., $Z1_{fail}$, but also $Z2_{fail}$ and $Z3_{fail}$). These instances occur most frequently, but are not always the focus for the coaching staff. In Fig. 10, two of these distinct models are compared with and without the least interesting class ($Z1_{fail}$) excluded (left and right panel, respectively).

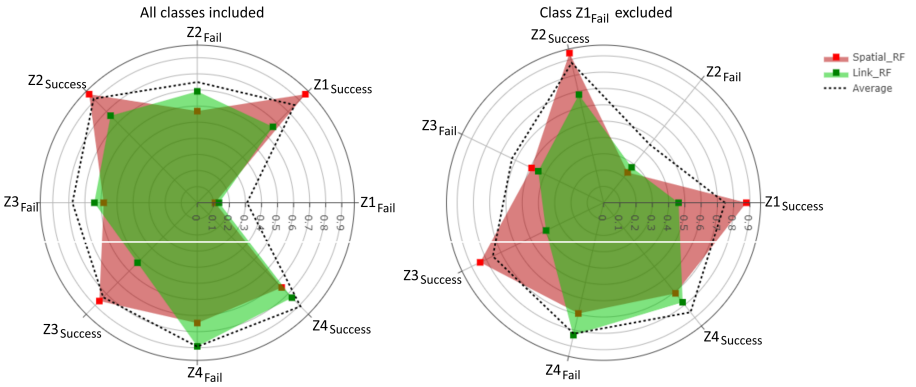

Fig. 10. A direct comparison of the *Spatial*- and *Link*- related models. Compared to including all classes (left panel), excluding the least interesting class gives a more nuanced insight into the relevant differences between the two models (right panel).

Looking at all classes, it stands out that $Z1_{fail}$ is predicted correctly more often than all other classes (see Fig. 10, left panel). As this is also the least interesting class, it clouds the accuracy for the other classes. After excluding $Z1_{fail}$ (see Fig. 10, right panel), it becomes clear that the *Spatial*-related model outperforms the *Link*-related model in the classes related to zone 4 ($Z4_{success}$ and $Z4_{fail}$). As these are attacking sequences starting from close to the opponent's goal, predicting these right is often less interesting than predicting the (successful) attacking sequences correct that start further away from the goal. Therefore, the use of all *Link*'s Dangerousity-related metrics is the most promising to examine how attacking sequences starting in different zones yield successful attacks.

7 Conclusion and future work

When selecting suitable Machine Learning models, the involvement of the expert is indispensable for evaluation, comparison and selection of models. This paper contributes to this overall goal by having proposed ML-ModelExplorer, involving the expert in the explorative analysis of the results of multiple multi-class classifiers. The design goals were deduced from typical, recurring tasks in the model evaluation process. In order to ensure a shallow learning curve, a combination of well-known visualisations together with some more advanced visualisations was

proposed. A user study was conducted where the participants were statistically significantly more efficient using ML-ModelExplorer than working on raw classification results from scikit-learn. Note, that the authors believe that scikit-learn is one of the most powerful libraries. However, for the recurring analysis of multiple models, approaches like ML-ModelExplorer can be powerful supplements in the toolchain of Machine Learning experts.

While the usefulness was experimentally shown, there is potential for further work. After performing the mentioned user stories, there is a possibility that no single model is sufficient to fulfil the requirements of the given application. This could be the case if the investigated models are strongly diverse in their decisions, which would lead to different patterns in their class confusions. So for example one model could have a very low accuracy on class C_1 but a high accuracy on class C_2, while another model acts vice versa. In this case, taking the diversity of the investigated models and classes into account, the composition of multiple models could be used to improve the classification performances through means of ensembles [5]. This would propose a useful addition to ML-ModelExplorer, especially in cases where the formulation of diversity of the classes and their respective confusions is not easily quantified [17]. Additionally, the mentioned user stories are not all encompassing. A scenario that is not covered, is the discovery of classes that seem to be not properly defined or labelled. Consequences could be the removal of that class, the separation into multiple classes or the merging with other classes. In order to compare the new class definitions with the existing one would require the comparison of differently sized confusion matrices.

ML-ModelExplorer enables domain experts without programming knowledge to reason about model results to some extent. Yet, there are open research questions, like how to derive concrete instructions for actions regarding goal-oriented model hyperparameter adjustment, i.e. letting the expert pose *what if*-questions in addition to *what is*-questions.

References

1. Alsallakh, B., Hanbury, A., Hauser, H., Miksch, S., Rauber, A.: Visual methods for analyzing probabilistic classification data. IEEE Transactions on Visualization and Computer Graphics **20**(12), 1703–1712 (Dec 2014)
2. Armatas, V., Yiannakos, A., Papadopoulou, S., Skoufas, D.: Evaluation of goals scored in top ranking soccer matches: Greek "superleague" 2006-08. Serbian Journal of Sports Sciences **3**, 39–43 (02 2009)
3. Bernard, J., Zeppelzauer, M., Sedlmair, M., Aigner, W.: Vial: a unified process for visual interactive labeling. The Visual Computer **34**(9), 1189–1207 (2018). https://doi.org/10.1007/s00371-018-1500-3
4. Chang, W., Cheng, J., Allaire, J., Xie, Y., McPherson, J.: shiny: Web Application Framework for R (2017), `https://CRAN.R-project.org/package=shiny`, r package version 1.0.5
5. Dietterich, T.G.: Ensemble methods in machine learning. In: Multiple Classifier Systems. pp. 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
6. Fawcett, T.: ROC graphs: Notes and practical considerations for researchers. Tech. rep., HP Laboratories (2004)

7. Frencken, W., Lemmink, K., Delleman, N., Visscher, C.: Oscillations of centroid position and surface area of soccer teams in small-sided games. European Journal of Sport Science **11**(4), 215–223 (2011). https://doi.org/10.1080/17461391.2010.499967
8. Goes, F.R., Kempe, M., Meerhoff, L.A., Lemmink, K.A.P.M.: Not every pass can be an assist: A data-driven model to measure pass effectiveness in professional soccer matches. Big Data **7**(1), 57–70 (3 2019). https://doi.org/10.1089/big.2018.0067
9. Goes, F.R., Meerhoff, L.A., Bueno, M.J.O., Rodrigues, D.M., Moura, F.A., Brink, M.S., Elferink-Gemser, M.T., Knobbe, A.J., Cunha, S.A., Torres, R.S., Lemmink, K.A.P.M.: Unlocking the potential of big data to support tactical performance analysis in professional soccer: A systematic review. European Journal of Sport Science p. to appear (2020). https://doi.org/10.1080/17461391.2020.1747552
10. Holzinger, A., Plass, M., Kickmeier-Rust, M., Holzinger, K., Crişan, G.C., Pintea, C.M., Palade, V.: Interactive machine learning: experimental evidence for the human in the algorithmic loop. Applied Intelligence **49**(7), 2401–2414 (2019). https://doi.org/10.1007/s10489-018-1361-5
11. Huang, W., Song, G., Li, M., Hu, W., Xie, K.: Adaptive weight optimization for classification of imbalanced data. In: Intelligence Science and Big Data Engineering, Lecture Notes in Computer Science, vol. 8261, pp. 546–553. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
12. Inc., P.T.: Collaborative data science (2015), `https://plot.ly`
13. Inselberg, A.: The plane with parallel coordinates. The Visual Computer **1**(2), 69–91 (1985)
14. Jiang, L., Liu, S., Chen, C.: Recent research advances on interactive machine learning. Journal of Visualization **22**(2), 401–417 (2019). https://doi.org/10.1007/s12650-018-0531-1
15. Kautz, T., Eskofier, B.M., Pasluosta, C.F.: Generic performance measure for multiclass-classifiers. Pattern Recognition **68**, 111 – 125 (2017). https://doi.org/https://doi.org/10.1016/j.patcog.2017.03.008
16. Krause, J., Perer, A., Bertini, E.: Infuse: Interactive feature selection for predictive modeling of high dimensional data. IEEE Transactions on Visualization and Computer Graphics **20**(12), 1614–1623 (Dec 2014)
17. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning **51**(2), 181–207 (2003)
18. LeCun, Y.: The MNIST database of handwritten digits (1999), `http://yann.lecun.com/exdb/mnist/`
19. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**, 436–44 (05 2015). https://doi.org/10.1038/nature14539
20. Link, D., Lang, S., Seidenschwarz, P.: Real time quantification of dangerousity in football using spatiotemporal tracking data. PLoS ONE **11**(12), 1–16 (12 2016). https://doi.org/10.1371/journal.pone.0168768, `https://doi.org/10.1371/journal.pone.0168768`
21. Meerhoff, L.A., Goes, F., de Leeuw, A.W., Knobbe, A.: Exploring successful team tactics in soccer tracking data. In: MLSA@PKDD/ECML (2019)
22. Memmert, D., Lemmink, K.A., Sampaio, J.: Current approaches to tactical performance analyses in soccer using position data. Sports Medicine **47**, 1–10 (06 2016). https://doi.org/10.1007/s40279-016-0562-5
23. Park, C., Lee, J., Han, H., Lee, K.: ComDia+: An interactive visual analytics system for comparing, diagnosing, and improving multiclass classifiers. In: 2019 IEEE Pacific Visualization Symposium (PacificVis). pp. 313–317 (April 2019)

24. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
25. Polikar, R.: Ensemble based systems in decision making. IEEE Circuits and Systems Magazine (2006)
26. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2017), `https://www.R-project.org/`
27. Raschka, S.: Model evaluation, model selection, and algorithm selection in machine learning. CoRR **abs/1811.12808** (2018)
28. Rawat, W., Wang, Z.: Deep convolutional neural networks for image classification: A comprehensive review. Neural computation **29**(9), 2352–2449 (2017)
29. Ren, D., Amershi, S., Lee, B., Suh, J., Williams, J.D.: Squares: Supporting interactive performance analysis for multiclass classifiers. IEEE Transactions on Visualization and Computer Graphics **23**(1), 61–70 (Jan 2017)
30. Sacha, D., Sedlmair, M., Zhang, L., Lee, J.A., Peltonen, J., Weiskopf, D., North, S.C., Keim, D.A.: What you see is what you can change: Human-centered machine learning by interactive visualization. Neurocomputing **268**, 164–175 (2017). https://doi.org/10.1016/j.neucom.2017.01.105
31. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: In Proceedings of Visual Languages. IEEE Computer Science Press. pp. 336–343 (1996)
32. Theissler, A.: Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection. Knowledge-Based Systems **123**(C), 163–173 (May 2017). https://doi.org/10.1016/j.knosys.2017.02.023
33. Zhang, J., Wang, Y., Molino, P., Li, L., Ebert, D.S.: Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. IEEE Transactions on Visualization and Computer Graphics **25**(1), 364–373 (Jan 2019)