Explaining Learning Performance with Local Performance Regions and Maximally Relevant Meta-Rules

Ricardo B. C. Prudêncio[1] and Telmo de M. e Silva Filho[2]

[1] Centro de Informática, Universidade Federal de Pernambuco
rbcp@cin.ufpe.br
[2] Department of Engineering Mathematics, University of Bristol
telmo.silvafilho@bristol.ac.uk

Abstract. Identifying instances in a learning task that are difficult to predict is important to avoid critical errors at deployment time. Additionally, providing explanations for good or bad predictions of a model can be useful to understand its behavior and to plan how to improve it (e.g., by data augmentation in specific areas of instances). In this paper, we propose a method to provide explanations for a model's predictive performance based on the induction of meta-rules. Each meta-rule identifies a local region in the instance space, called Local Performance Region (LPR). The meta-rules are induced using a reduced number of attributes, in such a way that each LPR can be inspected by, e.g., plotting a pairwise attribute plot. Additionally, given a group of instances to explain (or eventually an individual instance), we propose a greedy-search algorithm that finds the subset of non-redundant LPRs that maximally covers the instances. By explaining the (in)correctness of model predictions, LPRs constitute a novel use of meta-learning and a novel application in explainable AI. Experiments show the usefulness of LPRs while explaining inaccurate class predictions of Random Forest in a benchmark dataset, demonstrating a special case of LPRs, called Local Hard Regions (LHRs).

Keywords: Meta-learning, explainability, rule learning

1 Introduction

Predicting the performance of Machine Learning (ML) algorithms is an important task to support algorithm selection and to understand the limits of each algorithm of interest. This task has been treated by meta-learning [2] as another supervised learning task. In this approach, a set of training meta-examples is produced from experiments performed to evaluate a set of algorithms on a set of learning problems of interest. A meta-learner is then built to predict algorithm performance for new problems. In this paper, we are focused on the instance-level meta-learning approach [14,3,6], which is specific to predict algorithm performance for instances in a single learning problem of interest. So, before using a candidate model to predict an input instance, the meta-learner could be used to foresee if the model is actually adequate for that instance. Additionally, such an approach can be used to perform dynamic algorithm selection [5].

Previous work on meta-learning has focused on optimizing the predictive performance of meta-models, while neglecting explainability, which should be an important issue in meta-learning. In fact, a primary objective of meta-learning is actually to *understand* algorithm performance. However previous works are limited for example to learn meta-models that are directly interpretable, e.g., decision trees. The same challenges that motivated the topic of explainable ML [12] are also applied to meta-learning, and thus, further investigations are necessary to propose procedures for explaining meta-models. Hence, important practical questions such as when an algorithm fails could be answered more properly.

This paper proposes a new approach to explain learning performance by extracting meta-rules for a learned model in a dataset. Each meta-rule defines a Local Performance Region (LPR) in terms of subsets of instances and features for which a learned model has remarkable poor predictive performance (Local Hard Region – LHR) or strong predictive performance (Local Easy Region – LER). By producing meta-rules with one or two attributes, LPRs can be inspected in visual plots. As a second contribution, we proposed a greedy-search procedure to find the subset of LPRs which covers the maximal number of instances given by the user. Depending on the dataset complexity, the number of LPRs can be high but it is possible that different users aims at each time to explain specific groups of instances or even a single one. Thus, filtering non-redundant LPRs can be beneficial for satisfying each user's demand.

Current state-of-the-art explainers, such as LIME [16], act at the instance level and focus on which features were important for the class prediction given by the model. However this explanation is produced without taking into account the accuracy/error of the model's prediction. This is an important distinction between these explainers and our approach, which actually aims to explain performance, instead of individual predictions. This means that LPRs and LIME (or similar explainers) are complementary and can be used together for a full picture of the explained model's predictions.

Experiments were performed in the Statlog (Heart) classification dataset, in which 14 LHRs were identified for the Random Forest (RF) algorithm. Each LHR indicates a specific area in the instance space where RF produced relatively poor class probabilities. Each LHR could be explained by one or two attributes in the dataset, which could be inspected individually in the experiments. By adopting the maximal coverage search procedure, we identified 3 non-redundant LHRs which explained more than 80% of the RF's errors. In the experiments, we discussed how the LHRs can be used to explain the RF's performance for groups of hard instances or even for individual instances.

This paper is organized as follows. Section 2 presents the related work on meta-learning, followed by Section 3 in which we describe the proposed solutions. Section 4 in turn presents the performed experiments. Finally Section 5 concludes the paper with final considerations and future work.

2 Meta-Learning

Meta-learning predicts algorithm performance in a supervised way, usually applied to select algorithms at the dataset level [2]. In this case, a meta-learner is built from a set of meta-examples, in which each meta-example is related to a dataset and stores: (1) the characteristics describing the dataset, called meta-features (e.g., the number of attributes and examples, correlation between attributes, class entropy,...); (2) the candidate algorithm with best empirical performance for that dataset, usually estimated by cross-validation. A meta-learner model learned from a set of such meta-examples is a classifier that predicts the best algorithm for new datasets based on their meta-features. Alternatively, the meta-target stored in the meta-example can be the empirical performance estimation of a single algorithm. In this case, the meta-learner is a regression model which predicts the algorithm performance for new datasets.

In the instance-learning approach, meta-models are adopted to perform algorithm selection for each instance in a task of interest [3]. Thus, each meta-example is related to a single instance in a dataset. The meta-features in turn can be the original features of the instance or other features related to the models, like model confidence degree. The meta-target usually indicates the best algorithm to predict that instance. Alternatively, the meta-target can be a loss measure specific for an algorithm (e.g., 1|0 loss, absolute error,...). In this case, the meta-learner predicts the loss measure for each instance given as input.

Instance-level meta-learning has been adopted to perform dynamic selection of models in literature [5], where a meta-learner is used to select a candidate algorithm for each instance to classify [11,14,9,6,14]. Although related, our objectives are different (and complementary). Instead of identifying areas of competence for a query instance to classify, we aim to find all the local areas in instance space where a given model presented poor performance. Obviously such information can be used in dynamic classifier selection, for instance by discarding a model if the instance belongs to a known hard area for that model.

Finally, the current work is related to previous work dedicated to measure instance hardness [17]. Such previous work ranges from measuring the performance of a single model or a pool of models for each instance [17,15], instance hardness based on item response theory [10,4,13] and the use of data complexity measures [1]. The main focus of the previous work is to detect individual instances or groups of instances that are difficult to predict. Our work shares the objective of detecting hard areas in the instance space. However our proposed methods provide mechanisms to explain the instance hardness for a given model, which is not deeply investigated in literature.

3 Finding Local Performance Regions in Instance Space

In this paper, we propose an original method to explain algorithm performance for instance-level meta-learners. The main objective is to identify regions in the instance space that explain model performance. Hence the general question

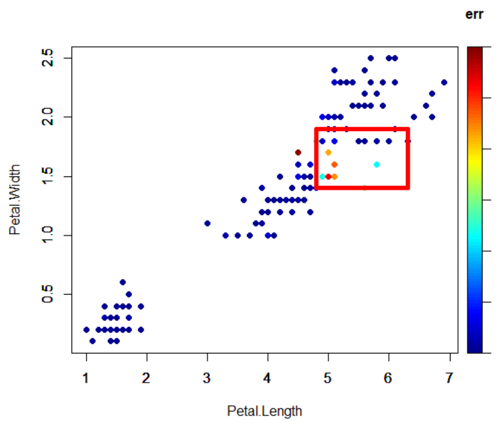


Fig. 1: Local Hard Region (LHR) in the Iris Dataset.

addressed in the paper is to know when a learned model fails or succeeds in its prediction. For this, we propose a method to detect Local Performance Regions (LPRs) and their special cases Local Hard Regions (LHRs) and Local Easy Regions (LERs). These local regions are subsets of instances in a dataset, defined by rules involving a reduced number of attributes (e.g., one or two), in such a way that model performance can be inspected in one or two dimensional scatter plots with the chosen attributes.

As a motivating example, consider the Iris dataset, which has four attributes and three classes. We train a Random Forest (RF) model and we aim to explain when it fails in its predictions. We start by assuming a bad prediction when the correct class probability predicted by RF is lower than 0.9 (or error greater than 0.1). This is obviously a strict criterion for defining that a model had a bad performance but notice that the Iris dataset is a very simple classification task. By using such criterion, we observed 10 hard examples, where 4 instances belong to the Virginica class and 6 belong to the Versicolor class. Figure 1 presents a possible LHR (the red frame) for inspecting the hard examples, using the Petal Length and the Petal Width attributes. This LHR covers 14% of the Iris dataset, with average RF error of 0.12. Additionally it covers 9 out of 10 hard examples, i.e., a single 2-D plot could explain 90% of the poor predictions of the RF model.

In this paper, we propose a method to find LPRs by learning discriminating meta-rules to explain the model performance (see Algorithm 1). Additionally, as several LPRs can be found for a single dataset and model of interest, we proposed a greedy-search method to select a reduced subset of LPRs to explain a group of instances given by the user. This is done by searching for the subset of LPRs that maximizes the total relevance for the given examples (see Algorithm 2). This algorithm can be eventually used to explain the (in)correctness for a single instance, by selecting a single LPR for that instance.

3.1 Definitions

Our work is focused on learning tasks for which we aim to derive explanations for the performance of a learned model. A task can be defined in terms of a set of predictor attributes $\{a_i\}_{i=1}^{m}$, where each attribute a_i has a domain A_i. Let $\mathcal{M}$ be a learned model and let D be a test dataset to collect the predictions and evaluate $\mathcal{M}$. For each instance $d \in D$, we collect the model's performance according to a chosen function: $L(\mathcal{M}, d)$.

An LPR in our work is defined by a meta-rule r that considers in its head a pair of attributes a_i and a_j and predicts the performance of model $\mathcal{M}$:

$$r : per = e \text{ when } a_i \in A_i^r \ \& \ a_j \in A_j^r,$$

where $A_i^r \subseteq A_i$ and $A_j^r \subseteq A_j$ are subsets of each attribute domain. We adopted two attributes in the LPRs for two reasons: (1) to provide simple explanations; and (2) to inspect the meta-rule in an LPR plot of two dimensions. Notice that a meta-rule with one antecedent is a special case of LPR in which $A_i^r = A_i$.

The model performance e returned by the meta-rule can be estimated as average $L(\mathcal{M}, d)$ for the instances where the meta-rule's conditions apply:

$$e(\mathcal{M}, r, D) = \frac{1}{n_r} \sum_{d \in D_r} L(\mathcal{M}, d), \tag{1}$$

where $D_r \in D$ is the subset of instances covered by r and $n_r = |D_r|$. The meta-rule coverage is the proportion of examples covered by the meta-rule:

$$Cov(r, D) = \frac{n_r}{n}, \tag{2}$$

where $n = |D|$.

3.2 Finding LPRs

Algorithm 1 shows the steps to find LPRs. The performance function $L(\mathcal{M}, d)$ can be chosen to measure model *incorrectness*, i.e., lower values are better, in which case Algorithm 1 outputs LHRs. Alternatively, if the user wishes to obtain LERs, $L(\mathcal{M}, d)$ can be chosen to measure model *correctness*, such that the higher its values, the better. As an example, the LHR presented in Figure 1 for the Iris problem is defined by the meta-rule:

$$r : per = 0.12 \text{ when Petal Length} \in [4.5; 6.3] \ \& \text{ Petal Width} \in [1.4; 1.8].$$

A number of $n_r = 21$ instances are covered by this meta-rule and hence coverage $Cov(r, D_{iris}) = \frac{21}{150}$. The average incorrectness returned by training the RF model for these 21 instances is 0.12 (i.e. $e(\mathcal{M}, r, D_{iris}) = 0.12$), with the chosen performance function being $L(\mathcal{M}, d) = 1 - p(y|d)$, where $p(y|d)$ is the probability given by $\mathcal{M}$ to the correct class y of instance d.

As mentioned above each LPR is related to a pair of attributes in the dataset. Given a pair of attributes a_i and a_j, a meta-dataset D_{ij} is produced where those attributes are used as predictors and the performance $L(\mathcal{M}, d)$ of model $\mathcal{M}$ for every instance in the dataset is used as the target attribute. A rule learning algorithm is applied on D_{ij} to derive a set of candidate meta-rules to predict the model performance based on a_i and a_j.

For LHRs, each candidate meta-rule is returned when the corresponding model *incorrectness* is greater than a pre-defined threshold ($thrE$), that is if $e(\mathcal{M}, r_k, D) > thrE$. In this case, the meta-rule actually distinguishes a bad performance region. We highlight that several LHRs can be found for a single attribute pair, depending on the value of $thrE$. The lower the value of this parameter, the higher the number of LHRs found by the algorithm. Obviously, $thrE$ has to be set according to the application domain, based on the model error that is tolerated. Naturally, the opposite happens for LERs, where a candidate meta-rule is returned when the corresponding model *correctness* is greater than the pre-defined threshold. Thus, as mentioned above, one must choose a suitable performance function in each case.

For LHRs, for example, the performance function $L(\mathcal{M}, d)$ can be chosen as any instance-wise error metric, such as a binary error function (1 if prediction was wrong, 0 otherwise), proper scoring rules (Brier score, log-loss, ...), or the complement of the probability predicted for the actual class of the instance, which we employ in our experiments in Section 4. Whichever metric is chosen, $L(\mathcal{M}, d)$ needs to be calculated on a dataset with available ground truth. However, our approach also allows for a trained meta-learner, which can be used to estimate $\hat{L}(\mathcal{M}, d)$ for new instances, meaning LPRs can be found to explain when the meta-learner thinks that $\mathcal{M}$ will predict poorly or not.

3.3 Finding Maximally Relevant LPRs

In this section we describe the method proposed to select a reduced subset $R \subseteq RuleSet$ to explain the set of examples D_u provided by the user. For example, D_u could be the set of all instances in D for which the model $\mathcal{M}$ returned errors greater than the threshold $thrE$, i.e. $L(\mathcal{M}, d) > thrE$. Alternatively the user could inform specific groups of instances to explain (e.g., all instances that are female and the model returned bad predictions). The proposed method is relevant when there are correlated predictor attributes in a dataset, in such a way that meta-rules extracted by the method proposed in previous section can be redundant.

The proposed method is closely related to [7], which proposed a greedy-search method to find 2-D plots of features to visualize outliers in a dataset. Our work adapted this idea to find subsets of LPRs that cover the instances provided by the user. The objective function $TotalRel(R, D_u)$ is based on a function that measures the relevance of R for each instance $d \in D_u$. The function considers the most relevant LPR that covers the example d:

$$Rel(\mathcal{M}, d, R) = \max_{r \in R} e(\mathcal{M}, r_k, D) \times Cov(r, d), \tag{3}$$

Algorithm 1: Find Local Performance Regions.

```
Input: Model $\mathcal{M}$; $n \times m$ test dataset $D$, in which $n$ is the number of instances
       and $m$ is the number of attributes; threshold $thrE$ for predicted model
       performance; and loss function $L(\mathcal{M}, d)$.
   /* Test model $\mathcal{M}$ on $D$ to collect the performances per (a $n \times 1$
      vector).                                                                 */
1  $\mathbf{per} \leftarrow (\mathbf{L}(\mathcal{M}, \mathbf{d}) \mid \mathbf{d} \in \mathbf{D})$
   /* Initialize the set of meta-rules with empty set.                         */
2  $RuleSet \leftarrow \{\}$
3  for each attribute pair $(a_i, a_j)$ do
       /* Produce a meta-dataset from $(a_i, a_j)$ and per.                     */
4      $D_{ij} \leftarrow$ Dataset built using the attributes $a_i$ and $a_j$ as predictors and
        model performance $per$ as the target attribute
       /* Extract meta-rules.                                                   */
5      $RuleSet_{ij} \leftarrow$ Rules extracted by applying a rule learning algorithm on $D_{ij}$.
       /* Select the meta-rules based on threshold $thrE$.                      */
6      for rule $r_k$ in $RuleSet_{ij}$ do
7          if $e(\mathcal{M}, r_k, D) > thrE$ then
8              $RuleSet \leftarrow RuleSet \bigcup r_k$
Output: $RuleSet$
```

where $Cov(r, d) = 1$ if the rule r covers the example d and it is 0, otherwise. The relevance function is useful to distinguish among several LPRs that compete to cover the example d. For example, the most relevant LHR for a hard instance is the one that has the highest predicted error. Finally the objective function for a subset R is defined as the total relevance:

$$TotalRel(R, D_u) = \sum_{d \in D_u} Rel(\mathcal{M}, d, R) \tag{4}$$

Algorithm 2 is a greedy-search algorithm that starts with an empty set of selected meta-rules and iteratively includes a candidate meta-rule that obtains the marginal total relevance for the user examples (see lines 6 and 7). A meta-rule is added at each iteration until a number b of meta-rules is selected.

3.4 Explaining LPRs Using Meta-Features

Algorithm 1 finds meta-rules defined using the features available in the dataset. As a complementary analysis, one could add an additional explanation layer to describe each LPR in terms of meta-features extracted from the data in the LPR. For example, instances in the minority class can be more frequent in a given LPR. If the model is biased to respond well for the majority class, the predictive model performance will be low for that region. Also, a classification task can be linear for certain regions of instances while more complex for other instances. In such analysis, the instances' features cause (high or low) data complexity, which in turn can explain the model performance in a different perspective.

Algorithm 2: Find the Meta-Rule Subset to Maximize the Total Relevance for User Examples

```
Input: $RuleSet$, set of meta-rules extracted by Algorithm 1; budget $b$, number
       of meta-rules to select; dataset $D_u$, set of user examples to cover.
   /* Initialization */
1  $candR \leftarrow RuleSet$ /* Subset of candidate meta-rules to select */
2  $selectR \leftarrow \{\}$ /* Subset of selected meta-rules */
3  $totalRel = 0$ /* Total relevance for the user examples */
   /* Select $b$ meta-rules, by maximizing marginal relevance for the
      user examples at each iteration */
4  while $size(selectR) < b$ do
       /* Compute the marginal relevance score of each candidate
          meta-rule */
5      for $each\ r_k \in candR$ do
6          $auxR \leftarrow selectR \bigcup r_k$
7          $margRel[r_k] \leftarrow TotalRel(auxR, D_u) - totalRel$
       /* Find the meta-rule in candidate set with maximal marginal
          relevance score */
8      $r^* \leftarrow argmax_{r_k \in candR} margRel[r_k]$
       /* Update the selected and candidate meta-rule sets */
9      $selectR \leftarrow selectR \bigcup r^*$
10     $candR \leftarrow candR \setminus r^*$
       /* Update the total relevance of the selected meta-rule set */
11     $totalRel = totalRel + margRel[r^*]$
Output: selectR
```

Different candidate meta-features are available in literature to analyze data complexity at the instance level, including metrics of feature relevance, linearity-based measures, class balancing measures, neighborhood-based measures, among others [1,17]. In the end-user perspective, meta-rules defined using the original features can be easier to interpret. In the data analyst perspective, identifying relevant meta-features in a LPR can be useful to understand model performance also considering the eventual biases the model has and the consequences to deal with the instances belonging to specific regions in the instance space.

4 Experiments

In this section we present an experiment to produce explanations for RF applied to the Statlog(Heart) dataset. This dataset is a binary classification task, containing 270 instances and 13 predictor attributes. In the experiments, we performed 10-fold cross-validation to collect the prediction errors returned by RF. For each instance d we computed the error obtained by RF:

$$L(\mathcal{M}, d) = 1 - p(y|d), \tag{5}$$

where y is the true class label of instance d and $p(y|d)$ is the predicted class probability returned by RF. In the experiment, we adopted the randomForest package in R, with default parameters.

The errors obtained by RF in the Heart dataset had an average value of 0.27. The distribution is skewed and the RF errors tend to be low. About 60% of errors were lower than average. More extreme errors (e.g., greater than 0.8) were observed for 2.9% of the instances.

4.1 Finding LHRs: Algorithm 1

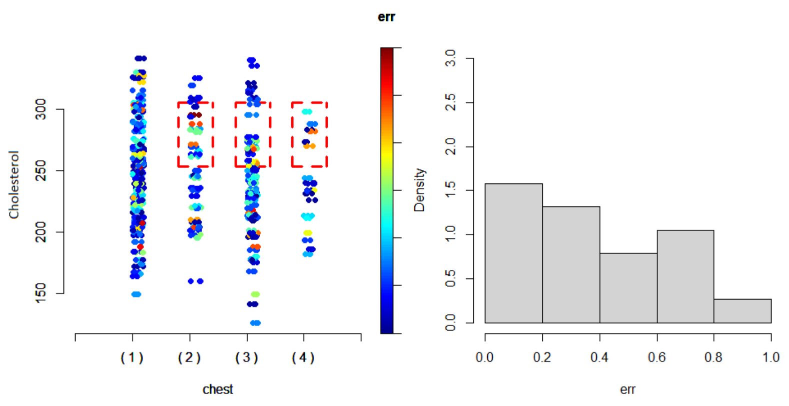
For finding the LHRs (Algorithm 1), we adopted the rpart package in R for rule induction. Initially a decision tree is induced using D_{ij}. Then the rpart.rules function is used to extract the meta-rules from the induced decision tree. All parameters of rpart were defined as the default values, apart from the *minbucket* parameter, which controls the coverage of instances in each terminal node. In our experiments, minbucket was set to 5% of the instances, in such a way that each returned LHR has a minimum level of representativeness in the dataset (i.e., $Cov(r, D) > 0.05$). This is to avoid very small LHRs that pinpoint single very hard instances. When few instances are present in a LHR, the rules can have a lower reliability. In future work, we aim to address this challenge by adopting data augmentation procedures to fill in the instance space with more instances, and potentially extracting more reliable LHRs.

Finally, in order to select the LHRs we adopted $thrE = 0.3$, which is about 10% greater than the average error. Table 1 shows the meta-rules returned by Algorithm 1. A total number of 14 meta-rules were induced. Notice that some extracted rules have only one variable in their condition statements as less relevant attributes were discarded by the rule learning algorithm. The predicted errors associated to these meta-rules ranged from 0.31 to 0.38, while the coverage values ranged from 14% to 39%. Rules are ordered by predicted error. Figure 2(a) shows the hardest LHR (Meta-rule #1, $err = 0.38$). Figure 2(b) presents the distribution of errors for the examples associated to this meta-rule, which, as expected, tend to be higher.

4.2 Filtering Meta-Rules: Algorithm 2

In this section, we discuss the results obtained by Algorithm 2 to maximize the coverage of hard examples. In this experiment, we defined a budget of 7 rules to select. The set of examples D_u was composed by all instances for which RF returned an error greater than 0.3 (i.e., the same threshold $thrE$ adopted in Algorithm 1). A total number of 98 instances were considered as hard examples in this set.

The 7 selected meta-rules were sufficient to explain most hard examples in D_u (coverage higher than 0.90). Table 2 presents the top three meta-rules filtered by Algorithm 2, which covered more than 80% of the hard examples. Meta-rule #12 (Figure 3) covers alone 16% of the given examples. The rule's conditions bring contradictory signals about the risk of heart disease, which make the LHR



(a) (b)

Fig. 2: LHR for Meta-rule #1 and histogram of errors. Color represents the RF error, ranging from 0 (dark blue) to 0.6 (dark red).

challenging for RF to provide good predictions. In order to understand this conflict, we first discuss the roles of each attribute to predict the disease class. First, in the Heart dataset the patient's pressure is alone an indicator that suggests a higher risk of heart disease. In turn, the attribute electrocardiographic has three distinct values (left-ventr-hyperthophy, normal and wave-abnormal), but only the value left-ventr-hyperthophy is clearly related to a high risk of heart disease. Hence, the pressure condition in Meta-rule #12 indicates a higher chance of disease, while the condition on electrocardiographic indicates a lower risk of disease. These conflicting signals cause high errors for the RF algorithm in both classes. The average RF error in this LHR was 0.32 for instances in the absence class and 0.33 for instances in the presence class.

Meta-rule #11 defines an LHR with medium-to-high values of cholesterol, with a prevalence of the presence class (56%) over the absence class (44%). The class entropy in this LHR causes a difficulty for the RF algorithm, with higher errors in this case for the absence class. Finally, Meta-rule #4 is similar to Rule #12 in the sense that it also brings conflicting conditions. The value 'upsloping' of the slope attribute is also an indicator of lower risk of disease, while high pressure is an indicator of higher risk. The class imbalance inside LHR #4 is higher than the other LHRs, with a prevalence of the absence class (75%). In this case, average RF error was higher for the presence class (0.38).

4.3 Explaining Single Instances

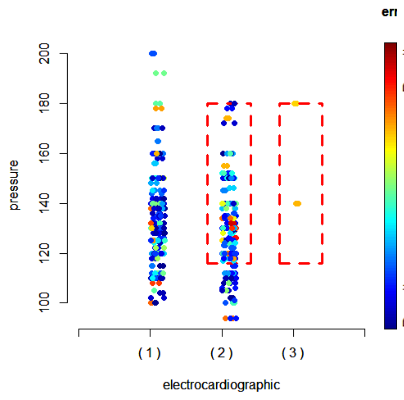
Algorithm 2 can be adopted to choose an LPR to explain a single instance. This is done by setting D_u as the instance of interest. In this case, Algorithm 2 will

Table 1: Meta-rules identified by Algorithm 1

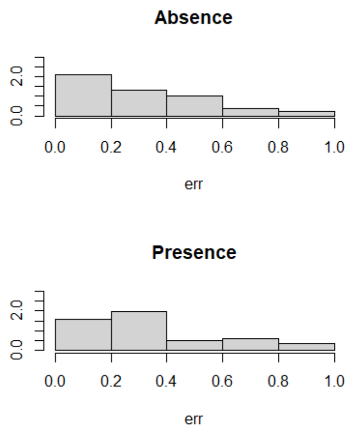
N.	Meta-Rule	Cov.
#1	per = 0.38 when cholesterol is 253 to 305 & chest is not asymptomatic	14%
#2	per = 0.38 when heartRate >= 134 & cholesterol is 253 to 301	20%
#3	per = 0.37 when cholesterol is 253 to 305 & oldpeak < 0.95	18%
#4	per = 0.36 when pressure is 116 to 135 & slope is upsloping	21%
#5	per = 0.36 when heartRate >= 162 & oldpeak < 0.05	18%
#6	per = 0.36 when pressure is 130 to 135	15%
#7	per = 0.35 when heartRate is 134 to 150	17%
#8	per = 0.35 when heartRate >= 162 & chest is asymptom. or atyp-angina	18%
#9	per = 0.34 when pressure is 116 to 135 & oldpeak < 1.45	30%
#10	per = 0.34 when pressure >= 116 & cholestoral is 253 to 305	28%
#11	per = 0.34 when cholesterol is 253 to 305	32%
#12	per = 0.33 when pressure >= 116 & electrocardio is normal or waveabn	39%
#13	per = 0.31 when heartRate >= 162	34%
#14	per = 0.31 when pressure >= 116 & cholesterol < 225	24%

Table 2: Meta-rules filtered by Algorithm 2

Selected Meta-Rule	Marginal Incrimination	Marginal Coverage	Absence (%)	Presence (%)	Avg. Error of Absence Class	Avg. Error of Presence Class
#12	0.16	0.49	61%	39%	0.32	0.33
#11	0.09	0.25	44%	56%	0.35	0.32
#4	0.03	0.07	75%	25%	0.34	0.38



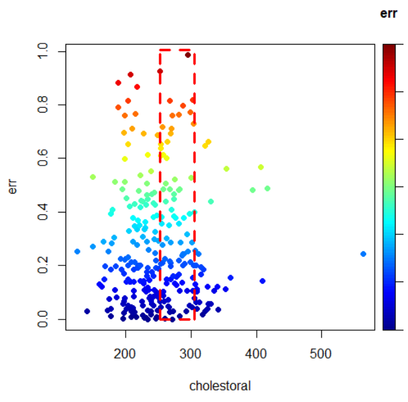
(a)



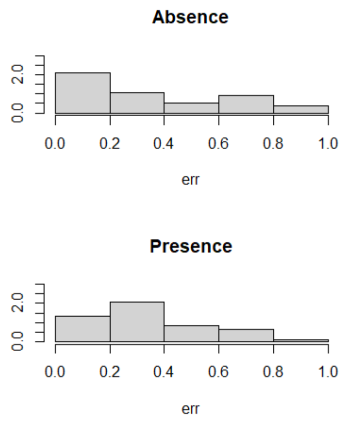
(b)

Fig. 3: LHR for the Rule #12 and histograms of errors per class.

return the most relevant LPR that covers that instance. In order to illustrate, we choose an instance for which RF returned a very high error. Specifically, we inspect the RF performance for an old man (77-years old), who has high pressure (125) and high cholesterol (304). This patient has heart disease (presence class), which could be expected by considering these indicators in isolation. The error

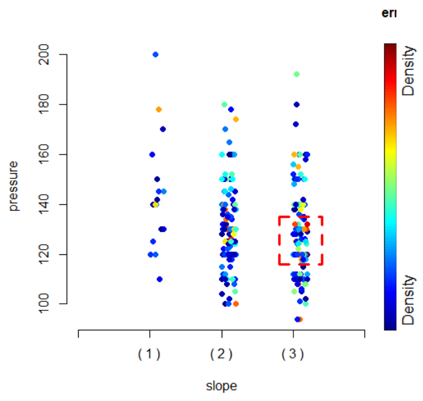


(a)

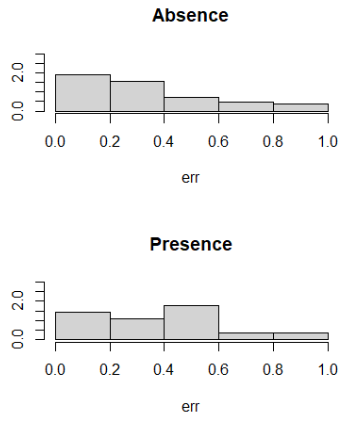


(b)

Fig. 4: LHR for the Rule #11 and histograms of errors per class.



(a)



(b)

Fig. 5: LHR for the Rule #4 and histograms of errors per class.

returned for RF for this patient was 0.73, which is actually a very high error. Algorithm 2 returned for this patient the Meta-rule #3 in Table 1. Again, this meta-rule has conflicting conditions associated to the risk of heart disease. The oldpeak attribute measures the heart stress during exercise, and low values indicate low risk of disease. In fact, in the Heart dataset, when oldpeak < 0.95, 106 instances belong to the absence class and 39 belong to the presence class, which corresponds to a low probability of 0.27 for the disease. On the other hand, when oldpeak >= 0.95, 44 instances belong to the absence class, while 81 belong to the presence class, i.e., probability of disease is 0.64.

By considering the cholesterol between 253 and 305, there are 38 instances in the absence class and 47 instances in the presence class, i.e, probability of disease is 0.55. Otherwise, 112 instances belong to the absence class and 73 instances in the presence class, i.e., probability of disease is 0.64. High cholesterol suggests high risk of disease. Although the patient of interest has characteristics that suggest a high risk of disease, he also has an exam result (oldpeak attribute) that suggests the opposite (oldpeak is equal to zero for this patient). These conflicting indicators make this patient a hard instance to be predicted by RF.

4.4 Baselines

Finally we evaluate two natural baselines that could be adopted to find LPRs:

– Baseline 1: a single decision tree is learned using all attributes and the rules which matched the threshold $thrE$ are returned. This baseline tends to produce more complex rules as no limit is defined in the number of attributes in each rule. Some rules for instance may not produce an LPR that can be plotted in 2D space;

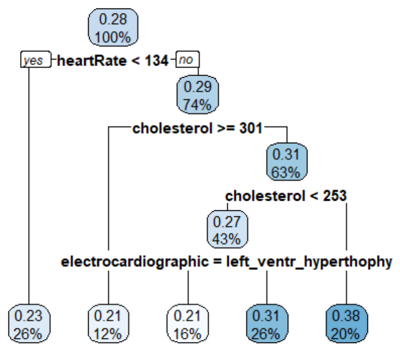
– Baseline 2: a single decision tree is learned using all attributes but limiting the maximal depth to 2, in such a way that the derived rules have at most 2 attributes. As our proposed method, this baseline produces simple rules.

The same threshold value $thrE = 0.3$ is adopted in these baselines. Figure 6(a) presents the decision tree induced in Baseline 1. Two meta-rules can be extracted from this decision tree by considering the thresholds, covering about 65% of hard examples. This baseline could extract a hard area with model error of 0.38, which is interesting for our purposes. However the returned meta-rules are more complex as they use three attributes.

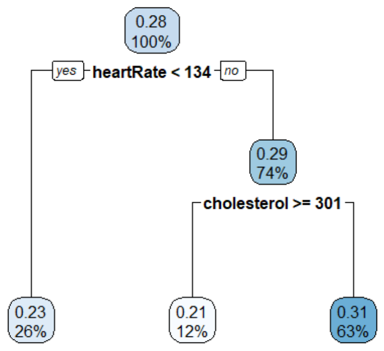
Figure 6(b) presents the decision tree induced in Baseline 2, which is similar to Baseline 1, but limiting the depth. A single meta-rule is returned in this decision tree, covering 75% of the hard examples. The proposed method to find LHRs provided alternative explanations that covered the remaining hard examples. Additionally the proposed method could find harder hard areas (e.g., Meta-Rule #1 in Table 1, which had a model error of 0.38). The proposed method is more flexible than the baselines since it finds LPRs in a pairwise way over the attributes. As a consequence, it can find more diverse explanations for the instances provided by the user.

5 Conclusion

In this paper we proposed an original method to find local areas in the instance space that explain model performance. The proposed method is based on learning meta-rules to predict model performance. By using a reduced number of



(a)



(b)

Fig. 6: Decision tree induced by (a) Baseline 1 and (b) Baseline 2. Each node contains the model error prediction and the coverage of examples in percentage.

attributes, graphical explanations for the model errors are provided in the form of LPR plots. Experiments were performed in a benchmark dataset to explain the bad predictions of random forest. The proposed method could return a variety of meta-rules to explain the bad RF predictions. Additionally, the proposed greedy-search algorithm could select a reduced number of non-redundant meta-rules to cover most of the hard instances.

As future work, we intend to investigate other rule learning algorithms or subgroup discovery methods, to identify LPRs. Additionally experiments will be performed in more benchmarks and algorithms, extracting LHRs, as well as LERs. The proposed methods can be investigated as a component of a dynamic algorithm selection procedure. Also, it can be combined with data augmentation procedures in order to improve robustness of the model to the hardest instances identified in a problem of interest. Finally, the proposed methods can be applied to explain meta-models that predict instance hardness measures. In this case, the proposed methods could not only identify LHRs specific to a single model, but instance regions in a learning task that are intrinsically hard to predict.

Although we presented our proof of concept by finding LHRs for RF in a classification task, Algorithm 1 can easily be used to find LHRs and LERs for regression tasks, as long as a suitable performance function is chosen. Additionally, LERs can be used to explain feature importance, by checking which attributes appear more frequently in the meta-rules. Finally, it is possible to use our method with meta-rules defined by more than two features and still visualize the resulting higher-dimensional LPRs, using graphical tools such as parallel coordinates [8]. We intend to investigate all of these possibilities as future works.

References

1. Arruda, J.L.M., Prudêncio, R.B.C., Lorena, A.C.: Measuring instance hardness using data complexity measures. In: Cerri, R., Prati, R.C. (eds.) Intelligent Systems. pp. 483–497. Springer (2020)
2. Brazdil, P., van Rijn, J.N., Soares, C., Vanschoren, J.: Metalearning: Applications to Automated Machine Learning and Data Mining. Springer (2022)
3. Brazdil, P., van Rijn, J.N., Soares, C., Vanschoren, J.: Metalearning in Ensemble Methods, pp. 189–200. Springer (2022)
4. Chen, Y., Silva Filho, T., Prudêncio, R.B., Diethe, T., Flach, P.: β^3-irt: A new item response model and its applications. In: 22nd Intern. Conf. on Artificial Intelligence and Statistics. Proc. of Machine Learning Research, vol. 89, pp. 1013–1021 (2019)
5. Cruz, R.M., Sabourin, R., Cavalcanti, G.D.: Dynamic classifier selection: Recent advances and perspectives. Information Fusion **41**, 195–216 (2018)
6. Cruz, R.M., Sabourin, R., Cavalcanti, G.D., Ing Ren, T.: Meta-des: A dynamic ensemble selection framework using meta-learning. Pattern Recognition **48**(5), 1925–1935 (2015)
7. Gupta, N., Eswaran, D., Shah, N., Akoglu, L., Faloutsos, C.: Beyond outlier detection: Lookout for pictorial explanation. In: Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., Ifrim, G. (eds.) Machine Learning and Knowledge Discovery in Databases. pp. 122–138. Springer (2019)
8. Inselberg, A.: The plane with parallel coordinates. The Visual Computer **1**(2), 69–91 (1985)
9. Khiari, J., Moreira-Matias, L., Shaker, A., Ženko, B., Džeroski, S.: Metabags: Bagged meta-decision trees for regression. In: Machine Learning and Knowledge Discovery in Databases. pp. 637–652. Springer (2019)
10. Martínez-Plumed, F., Prudêncio, R.B.C., Martínez-Usó, A., Hernández-Orallo, J.: Item response theory in ai: Analysing machine learning classifiers at the instance level. Artificial Intelligence **271**, 18–42 (2019)
11. Merz, C.J.: Dynamical learning bias selection. In: Proceedings of Machine Learning Researchl Intelligence and Statistics. vol. R0, pp. 386–395 (1995)
12. Molnar, C.: Interpretable Machine Learning. 2 edn. (2022), `https://christophm.github.io/interpretable-ml-book`
13. Moraes, J.V., Reinaldo, J.T., Ferreira-Junior, M., Filho, T.S., Prudêncio, R.B.: Evaluating regression algorithms at the instance level using item response theory. Knowledge-Based Systems **240**, 108076 (2022)
14. Pinto, F., Soares, C., Mendes-Moreira, J.: Chade: Metalearning with classifier chains for dynamic combination of classifiers. In: Machine Learning and Knowledge Discovery in Databases. pp. 410–425. Springer (2016)
15. Prudêncio, R.B.C.: Cost sensitive evaluation of instance hardness in machine learning. In: Machine Learning and Knowledge Discovery in Databases. pp. 86–102. Springer (2020)
16. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should i trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1135–1144 (2016)
17. Smith, M.R., Martinez, T., Giraud-Carrier, C.: An instance level analysis of data complexity. Machine learning **95**(2), 225–256 (2014)