184.702 Machine Learning
Exercise 3.1: Metalearning

David Molnar
Student - 01326891
e01326891@student.tuwien.ac.at

Lukas Kathrein
Student - 01325082
e01325082@student.tuwien.ac.at

Daniel Füvesi
Student - 01326576
e01326576@student.tuwien.ac.at

ABSTRACT
Metalearning is a form of learning based on meta information, it could also be said, learning about learning. This technique helps to identify and generalize approaches which can be used in the field of machine learning to gain useful information about datasets and the behaviour of respective classifiers. This learning process should guide through the decisions of which classifiers to use and what outputs in terms of testing results like precision and recall can be used. In this short summary a java based framework will be presented which uses the WEKA machine learning library to extract features out of datasets and the best classifier out of several. This newly generated and extracted information will then be used as input for the last step, where cross validation with three classifiers is performed. At the end, results are compared and presented.

Keywords
Metalearning, WEKA

1. INTRODUCTION
Metalearning should help to better (pre) select algorithms, which have different properties and parameters, and guide the process of machine learning regardless of classification or regression tasks. This learning process also helps to guide and supply a frame around the selection process, as well as minimize the time consuming experimenting with parameters on algorithms which in the end do perform only poorly. Also should the overall decision be elevated to a more serious approach and yield a more transparent selection of algorithms and thus help in an overall professional discussion.

There is, not surprisingly, a lot of background information on this topic, is it the aim of many professionals in the industry to transparently (pre) select algorithms for datasets and eliminating the "gut-feeling" approach and "hinges" about parameterizing selected algorithms. At this point it is not beneficial to go to deep into the theory of metalearning, due to the fact that for this assignment the frame and procedure for the meta-learning framework which in the end should be presented was already given. The steps, which will shortly be presented and discussed, recline towards the Rice framework. As a first step several datasets have to be selected, which build the ground foundation for all following steps. Independent from the datasets, features from statistical analysis, information theory, descriptive or model-based fields are to be extracted. In parallel several classifiers with default settings perform their tasks and the best result gets reported back. This back reporting is the fourth step and is used to generate, with the extracted features from step two, a completely new dataset. This newly created dataset represents the actual metalearning basis on which, in a last step three newly classifiers try to label which classifier will produce the best result.

So the overall goal is to predict a classifier (or a family) which will perform very well on a specific kind of dataset with the meta information extracted.

2. APPROACH
As WEKA is a java based library it is clear, that java got used as underlying infrastructure. The datasets are loaded from the respective resource folder and converted into WEKA Instances (a wrapper class representing datasets, but also single instances in a dataset). Further these instances are passed to the feature extractors and in parallel to the classifiers. Also runs each feature extractor and each classifier on a seperate thread to minimize the duration the frameworks needs for one run. The classifiers use a train/test split approach of 80/20. Thread managing and joining threads together is done with java's executorservice and future objects. These futures are needed to get the results out of the calculations performed to then write them into a file which gets converted again into a WEKA instance. This meta information instance is then presented to the final classifiers which use cross validation and output their results.

3. DATASETS
Due to the fact, that datasets from previous assignments were allowed and that no further restriction was imposed upon the selection of datasets, the selection for fitting datasets was fairly easy. The UCI repository yields a fast amount and the experience from the last assignments did help in terms of eliminating certain sets. That some datasets were favoured over others was simply of practical nature, it was not the interest of the team to start of with unnecessary hard datasets and maybe get lost in details when there are others, more straight forward sets to use. In a first round several small datasets from classification problems were selected and built the ground truth for further implementations. One problem that did occur was, that not all UCI repository datasets link to a data folder and thus it was impossible to get hold of the (original) data. Also was it a first criterion to minimize possible preprocessing which might conflict with the used WEKA framework, because prior to this no

direct java implementation was done with this library. As it was clear that there are no problems whatsoever with the implementation more and more sets were added and the final selection can be seen in Table 1 with some basic characteristics extracted from each dataset.

Table 1. Used Datasets (instances, attributes, classes)

Dataset	# Instances	# Attributes	# Classes
Winequality-red	1,599	12	6
Winequality-white	4,898	12	7
leaf	340	16	30
Breast-cancer-wisconsin	699	10	2
Breast-tissue	106	10	6
Credit-screening	690	16	2
Eye detection	14,979	15	2
Fertility diagnosis	100	10	2
Glass-identification	214	10	6
indian liver patient db	583	10	2
Pittsburgh-bridges-v1	108	13	7
Pittsburgh-bridges-v2	108	13	7
pokerhand	25,010	11	11
seismic pumps	2,583	19	2
wine	178	13	3
Banknote Auth	1372	5	2
Connectionist Bench	208	60	2
Credit Card Clients	30000	24	2
Mushroom	8124	22	2
Letter Recognition	20000	16	26
Total	111,899		

Figure 1 shows the distribution of instances and attributes in form of a scatter plot. The majority of the datasets lies in the lower left corner with instances between 100 and a couple of thousands, with a number of attributes between 5 and 20. However, other datasets with higher number of instances and attributes have also been selected for a better generalisation of this task.

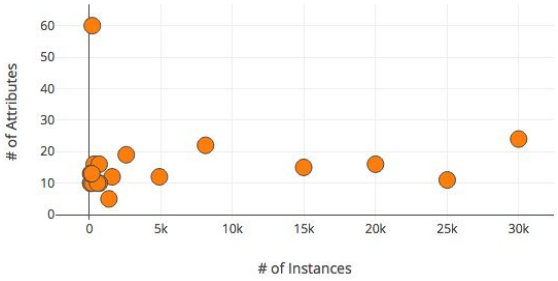

Figure 1: Scatter plot between number of instances and number of attributes of used datasets

3.1 Pre-processing

Target labels

In order to fully embrace the automated behaviour of the framework, two tiny pre-processing steps had to be implemented. Since WEKA requires the marking of the label class, the target column of each dataset has been renamed to either "label" or "class" so that an iteration through a high number of possible strings can be avoided (e.g. some datasets name their target column "score" or in worst case something domain-specific like "unique" or "poisonous"). This has been done manually, that hasn't been time-consuming due to the relatively low number of datasets.

Removal of "ID" columns

Before the datasets were fed into the framework, "ID" (also known as counter) columns were completely removed. This attribute consists of numeric values which obviously do not contribute to the relationship between data. To make sure that these values cannot be taken into consideration in successive steps by the framework, this attribute has been removed from the dataset it occurred in an automated way.

4. FEATURES EXTRACTED

As mentioned above, each dataset can be described by statistical, information theoretical, general or model-based features. These types of measures characterize the dataset in different ways. While statistical and descriptive measures identify simple characteristics among attributes, information theoretical features try to extract characteristics from the quantity of the information gain and the amount of uncertainty of the dataset. Properties of trained models on a particular dataset can also be used as an indirect form of characterization. These extracted features form the basis for the actual meta-learning step.

4.1 General features

Number of classes

This extractor obtains the number of possible classes for a dataset.

Number of features
Extracts the number of attributes (columns) of the dataset.

Number of instances
Extracts the number of samples in the dataset.

Proportion of missing values
Calculates the proportion of missing values within a dataset (e.g. 5%).

4.2 Statistical features

For each dataset the mean and the standard deviation (of mean) of the following four statistical measures have been calculated:

- Variance
- Kurtosis
- Skewness
- Correlation

These extractors are based on attribute to attribute comparisons and calculate the mean and standard deviation of the respective measures. As a result, eight new features were added to the resulting dataset describing the statistics of the datasets.

4.3 Information theoretical features

Entropy
Entropy of the dataset has been calculated using its well-known equation:

$$\mathrm{H}(X) = \sum_{i=1}^{n} \mathrm{P}(x_i)\,\mathrm{I}(x_i) = -\sum_{i=1}^{n} \mathrm{P}(x_i)\log_b \mathrm{P}(x_i)$$

4.4 Model-based features

REP Tree size
Not to forget about the model-based features, an REP Tree is trained in order to obtain model-based properties of the dataset. The size of the tree (i.e. number of nodes) got extracted and used for further training.

The extractors were executed in a parallelised manner, so that computation could be accelerated. To summarise up this section, a total of 15 attributes have been generated through these extractors.

5. CLASSIFIERS

Bayes Net
A base implementation of a bayes network which is able to learn from data instances based on several algorithms and quality measures. Data Structures used by the WEKA implementation are network structures, conditional probability distribution and more. The two most common algorithms used for the bayesian net are K2 and B, according to the implementation documentation.

In the proposed framework got the default constructor with its parameters used and not the also available bayesian network generator. The generator would allow a random network generation, but due to the information presented, only a default implementation got used.

The implementation of WEKA is unable to handle numeric classes this is why a conversion to nominal/categorical classes has to be made automatically to use the classifier. Also is this the sole representative of this family of classifiers which allows a broader coverage of algorithm families.

KStar
Being an instance based classifier, and the fact that in other assignments there was no chance to use one, the KStar implementation made it into the selection of classifiers used. As an instance based classifier this implementation classifies the test instance based on training instances which are similar to the one at hand. The similarity is defined by a default similarity function and uses internally entropy-based distance functions. The same as for the bayesian net, is this KStar algorithm the only representative for an instance based learner.

Multilayer Perceptron
This is the implementation of a neural network using back propagation to classify its instances. In the default construction of the class all nodes use a sigmoid activation function with the exception of the output nodes which become linear units without threshold. This is only the case when, as in many instances used here, the class is of numeric form. Also is this classifier the only one based on perceptrons used.

Random Forest
A basic implementation of a forest constructed by fully grown random trees. The default constructor yields a base classifier for bagging to random trees with a maximum number of iterations of 100.

Randomized Filter
WEKA allows a random filter classifier, which uses an arbitrary classifier on the dataset which gets passed through an arbitrary filter. The structure of the filter used is based exclusively on the training dataset, test instances will be processed by the generated filter but without changing the structure.

REPTree
As a second representative from the tree family this classifier, which can be used for regression and classification, is a fast decision tree learner and uses information gain as well as variance. Pruning of overfitted trees is done by reduce-error pruning (with backfitting). The algorithms implementation sorts numeric attributes once and missing values are split into corresponding pieces.

ZeroR Classifier
As last classifier and very basic implementation the 0-R classifier got selected. The selection and prediction is done as follows: numeric classes get predicted by the mean and nominal classes by mode.

6. META-INSTANCE CREATION

The creation of the dataset representing all datasets used as input with their meta information presented as well as the best achieving classifier are simply exported as a new instance which can be read again. The dataset consists of the 15 derived features plus the class label denoting the best classifier when using default parameters. Figure 2 shows a distribution of these classifiers.

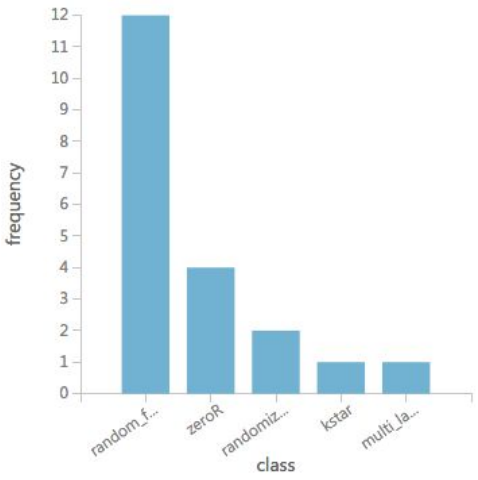

Figure 2: Distribution of best-performing default classifiers per dataset

As one can see in the figure, the Random Forest classifier occurs 12 times out of 20 instances. The reason for that could be the fact, that decision trees simply rely on if-else queries such that default settings yield better results than e.g. a multi-layer perceptron with default settings. Furthermore, there are two features that can be said to influence which default classifier performs the best. Figure 3 and Figure 4 both show boxplots how the number of features (attributes) and the proportion of missing values are connected with the default classifiers that perform best.

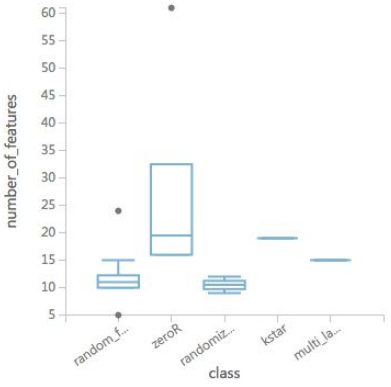

Figure 3: Relationship between number of attributes and best-performing default classifiers

It is not surprising that given a non-zero proportion of missing values, only a Random Forest, ZeroR or Randomized Filter wins due to the fact that Kstar and MLPs do not handle missing values well.

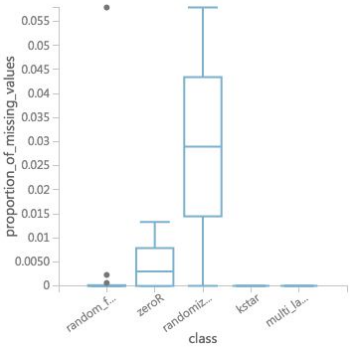

Figure 4: Relationship between proportion of missing values and best-performing default classifiers

Additionally, the linear correlations between the resulting features have also been measured. Unfortunately, the majority of the features do not correlate with each other well. However there are two pairs that particularly stand out. Figure 5 depicts how an increase of the number of classes also affects the entropy. Note that the axes represent the Pearson's correlation coefficients.

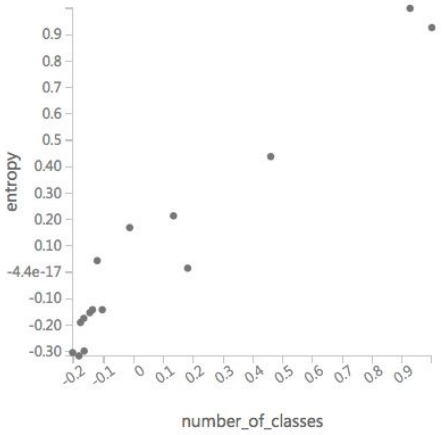

Figure 5: Linear correlation between Entropy and number of classes

An even higher correlation could be measured between the mean of kurtosis and mean of skewness. Figure 6 shows an almost perfect linear correlation. This proves that kurtosis and skewness often go hand in hand and that both of these features should be extracted when applying meta-learning.

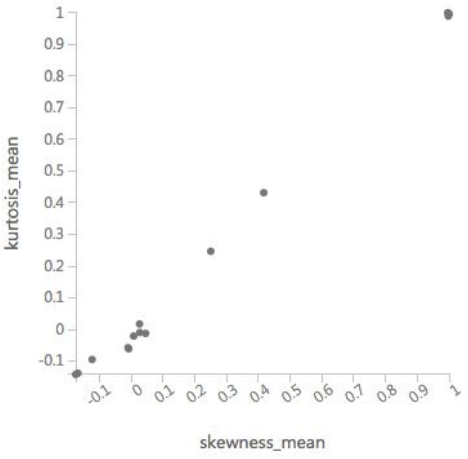

Figure 6: Linear correlation between Kurtosis (mean) and Skewness (mean)

7. CROSS-VALIDATION RESULTS

For the last step, three classifiers have been applied to the resulting dataset. Again, only default parameters were picked. Results:

- Random forest: 45% Accuracy
- Kstar: 35% Accuracy
- MLP: 40% Accuracy

As an additional experiment, a simple multi-class decision forest has been hyper-tuned and evaluated using 10-fold cross-validation. Figure 7 shows the confusion matrix. With a slight tuning, an accuracy of 60% could be reached.

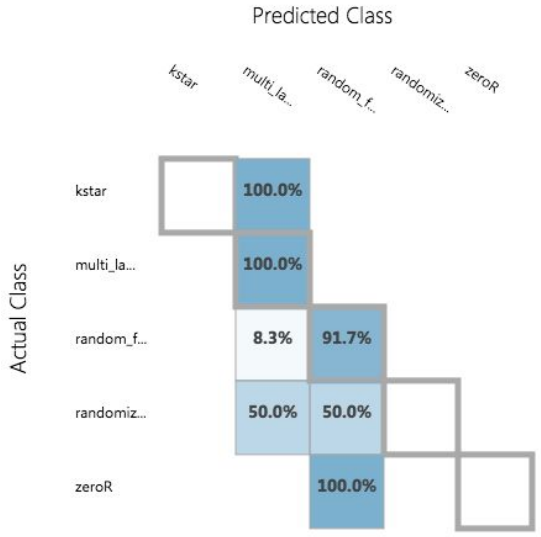

Figure 7: Confusion Matrix of a tuned decision tree applied on the meta-learning dataset

With additional feature selection, namely by reducing the features to only considering the number of features, proportion of missing values and the entropy, accuracy could be increased by an additional 10%. Note, that the feature selection has been done based on Mutual Information scores. Figure 8 shows the resulting confusion matrix, where the ZeroR instances have all been correctly classified.

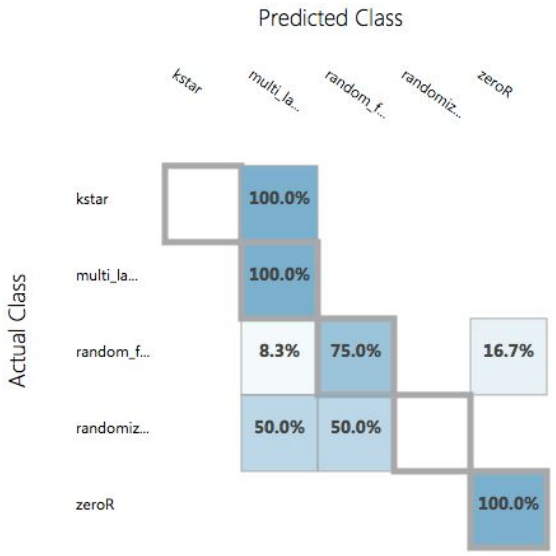

Figure 8: Confusion Matrix of a tuned decision tree applied on the reduced meta-learning dataset

8. SUMMARY

When drawing a conclusion it can be observed, that in order to build an accurate meta-learning model there is (as usual) a need for more data. Gathering more data in this case is easy, since there are hundreds of freely available datasets on the Internet. However, applying classifiers on these datasets do take a while even with default parameters. Looking at Table 1, there are only three relatively big datasets: Eye-detection (15k instances), Pokerhand (25k instances) and Client Credit Card (30k instances). Without these datasets, the whole framework finishes within 10-15 minutes on a 4-core CPU machine. Including these big datasets quadruples the runtime, which essentially means waiting one whole hour. One hour, for building a meta-learning dataset with only 20 datasets. Even if the calculation can be optimised and better distributed for example in the cloud, getting a meta-learning dataset with a decent size (100-200 instances) means a significant impact on the runtime. This however could potentially invalidate the main reason meta-learning is applied: to save time choosing the right algorithm and tuning parameters.

An alternative method could be to avoid the diversification of datasets, i.e. instead of choosing datasets with varying attributes and instance sizes (like in this study), only datasets with similar characteristics are picked. This approach has the advantage of focusing on a subset of problems one at a time. In case of small datasets this means that a lot of them can be gathered and fed into the framework without immensely impacting the runtime. The disadvantage is of course that the resulting model is inherently biased towards the information that characterize small datasets, so that the whole approach should be revisited for bigger and bigger datasets. Meta-learning is no perfect recipe for solving machine learning and obviously has some trade-offs. However, if one can successfully tackle its challenges and has the resources to build a model with high predictive power, a lot of time, anger and energy can be saved on the long run.