Evolving Decision Trees Using Oracle Guides

Ulf Johansson and Lars Niklasson

Abstract—**Some data mining problems require predictive models to be not only accurate but also comprehensible. Comprehensibility enables human inspection and understanding of the model, making it possible to trace why individual predictions are made. Since most high-accuracy techniques produce opaque models, accuracy is, in practice, regularly sacrificed for comprehensibility. One frequently studied technique, often able to reduce this accuracy vs. comprehensibility tradeoff, is rule extraction, i.e., the activity where another, transparent, model is generated from the opaque. In this paper, it is argued that techniques producing transparent models, either directly from the dataset, or from an opaque model, could benefit from using an oracle guide. In the experiments, genetic programming is used to evolve decision trees, and a neural network ensemble is used as the oracle guide. More specifically, the datasets used by the genetic programming when evolving the decision trees, consist of several different combinations of the original training data and "oracle data", i.e., training or test data instances, together with corresponding predictions from the oracle. In total, seven different ways of combining regular training data with oracle data were evaluated, and the results, obtained on 26 UCI datasets, clearly show that the use of an oracle guide improved the performance. As a matter of fact, trees evolved using training data only had the worst test set accuracy of all setups evaluated. Furthermore, statistical tests show that two setups, both using the oracle guide, produced significantly more accurate trees, compared to the setup using training data only.**

I. Introduction

Most data mining projects contain the generic process referred to as *predictive modeling*. A predictive model is a mapping, from an input vector consisting of attribute values, to a scalar output called the *target variable*. The overall purpose of such modeling is to be able to accurately predict target values from input attribute values. If the target variable is restricted to a predefined set of discrete labels (classes), the data mining task is called *classification*.

When using machine learning techniques, the predictive model represents patterns in historical data found by the specific algorithm. More technically, the algorithm uses a set of *training instances*, each consisting of an input vector $\boldsymbol{x}(i)$ and a corresponding target value $y(i)$ to learn the function $y=f(\boldsymbol{x};\theta)$. During training, the parameter values θ are optimized, based on a *score function*. When sufficiently trained, the predictive model is able to predict a value y, when presented with a novel (test) instance $\boldsymbol{x}$. Figure 1 below shows a schematic picture of predictive modeling.

U. Johansson is with the School of Business and Informatics, University of Borås, SE-501 90 Borås, Sweden. Email: ulf.johansson@hb.se

L. Niklasson is with the Informatics Research Centre, University of Skövde, Sweden. Email: lars.niklasson@his.se

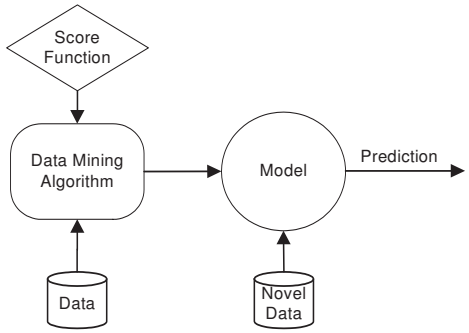

Figure 1: Predictive modeling.

Here, the data mining algorithm produces a model by somehow optimizing a score function on available training instances. Predictions on novel data (a *test set*) is then made, based on the model.

When performing predictive modeling, the most important criterion is *accuracy*, i.e., predictions made on novel data must be as correct as possible. Unfortunately, most high-accuracy techniques. like *artificial neural networks* (ANNs), *ensembles* or *support vector machines*, produce opaque models. Opaque predictive models make it impossible to follow and understand the logic behind a prediction, and on another level, for decision-makers to comprehend and analyze the overall relationships found. Clearly, there are domains, like the medical field, where this is unacceptable, making transparent models more or less mandatory. So, when models need to be interpretable (or even comprehensible) accuracy is often sacrificed by using simpler, but transparent models; most typically *decision trees* like C4.5/C5.0 [1] or CART [2]. This tradeoff between predictive performance and interpretability is normally called the *accuracy vs. comprehensibility tradeoff*.

As mentioned above, the most straightforward way of obtaining transparent models is, of course, to build a decision tree or induce a rule set directly from the training data. Another option is, however, to generate a transparent model based on a corresponding opaque predictive model. This process, which normally is called *rule extraction*, has been used mainly on ANN models; for a good survey see [3]. Rule extraction consequently produces another model which, in most cases, is used for the actual prediction. With this in mind, accuracy must be considered the prioritized criterion for rule extraction as well. Although this may seem almost obvious, most rule extraction techniques instead maximizes *fidelity*, i.e., how well the extracted model mimics the opaque.

At the same time, it is important to realize that the opaque model normally is a very accurate model of the relationship

978-1-4244-2765-9/09/$25.00 ©2009 IEEE

between input and target variables. One might even argue that an highly accurate opaque model often is a more correct representation of the data than the dataset itself. One example is that training instances misclassified by the opaque model may very well be atypical, i.e., learning such instances could reduce the generalization capability.

More importantly, the opaque model could also be used to generate predictions for novel instances with unknown target values, as they become available. Naturally, these instances could also be used by the rule extraction algorithm, which is a major difference compared to techniques directly building transparent models from the dataset, where each training instance must have a known target value. Despite this, all rule extraction algorithms that the authors are aware of use only training data (possibly with the addition of artificially generated instances) when extracting the transparent model.

We have previously argued that it could be advantageous for a data miner to also use test data together with predictions from the opaque model when performing rule extraction [4]. In that paper, we referred to test data inputs together with test data predictions from the opaque model as *oracle data*, with the motivation that the predictions from the opaque model (the oracle) were regarded as ground truth during rule extraction. Naturally, target values for test data are by definition not available when performing rule extraction, but often input values and predictions from the opaque model could be. With access to a sufficiently sized oracle dataset, the rule extraction algorithm could use either just the oracle data, or augment the training data with oracle instances. More generally, any data mining technique producing transparent models could potentially benefit from using different combinations of standard training instances, training instances with oracle predictions and oracle data during learning.

The overall purpose of this paper is to evaluate the use of high-accuracy models as oracles, guiding the extraction or induction of transparent models.

II. Background and Related Work

The overall goal of rule extraction is to produce a transparent model which is able to mimic the corresponding opaque model as well as possible, thus trying to keep an acceptable accuracy. Most well-known rule extraction algorithms are used to extract symbolic rules from trained neural networks, e.g., RX [5] and TREPAN [6]. Several papers have identified and discussed key demands on reliable rule extraction methods, see e.g., [3] or [7]. The most common criteria are: *accuracy* (the ability of extracted representations to make accurate predictions on previously unseen data), *comprehensibility* (the extent to which extracted representations are humanly comprehensible) and *fidelity* (the extent to which extracted representations accurately model the opaque model from which they were extracted). Accuracy and fidelity are measured as the proportion of instances classified correctly or identically to the opaque model, respectively. Comprehensibility is a rather complex criterion, but normally it is simply evaluated based on the size of the model.

We have previously suggested a rule extraction algorithm called G-REX (Genetic Rule EXtraction) [8]. G-REX is a *black-box* rule extraction algorithm, i.e., the fundamental idea is to view rule extraction as a learning task, where the target concept is the function learnt by the opaque model. Black-box techniques typically use some symbolic learning algorithm, where the opaque model is used to generate target values for the training instances. Black-box rule extraction techniques differ in the *representation language* used (i.e. the format of the extracted models) and exactly how the models are constructed, the *extraction strategy*. The most common families of representation languages are symbolic rule sets and decision trees.

Black-box rule extraction is, consequently, an instance of predictive modeling, where each input-output pattern consists of the original input vector and the corresponding prediction from the opaque model. From this perspective, black-box rule extraction becomes the task of modeling the function from (original) input attributes to the opaque model predictions, see Figure 2 below.

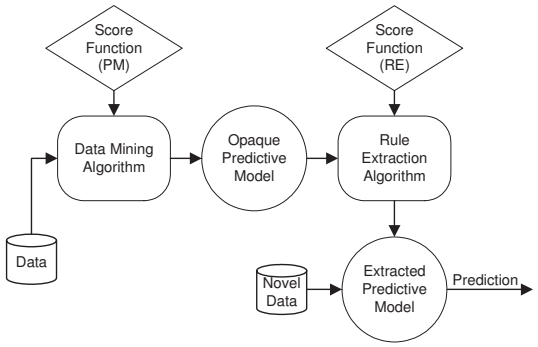

Figure 2: Black-box rule extraction.

One inherent advantage of black-box approaches is the ability to extract rules from arbitrary opaque models, including ensembles.

The extraction strategy used by G-REX is based on Genetic Programming (GP). More specifically, a population of, initially random, candidate models is evolved, using a score (fitness) function based on fidelity. During evolution the best (most fit) models are kept and combined using genetic operators to improve the fitness over time. After many iterations (generations) the best model (individual) is chosen as the extracted model.

One key property of G-REX is the ability to use a variety of different representation languages. G-REX has previously been used to extract, for instance, decision trees, regression trees, Boolean rules and fuzzy rules. Another, equally important, feature is the possibility to directly balance accuracy against comprehensibility (model size) by using a fitness function penalizing more complex models.

Lately, G-REX has been substantially modified, with the aim of becoming a general data mining framework based on GP; see [9]. For a summary of the G-REX technique and previous studies, see [10].

As mentioned above, the use of oracle data was suggested in [4], where the main result was that rules extracted using oracle data were significantly more accurate than both rules extracted by the same rule extraction algorithm (using training data only) and standard decision tree algorithms. It must be noted that the use of oracle data requires a sufficiently sized oracle dataset, i.e., the problem must be one where predictions are made for sets of instances rather than one instance at a time. The reason is, of course, that the same novel (unlabeled) instances used for actual prediction also are used by the rule extraction algorithm. Fortunately, in most real-world data mining projects, bulk predictions are made, and there is no shortage of unlabeled instances.

One example, where oracle data would not be available, is a medical system, where diagnosis is based on a predictive model built from historical data. In that situation, test instances (patients), would probably be handled one at a time. On the other hand, when a predictive model is used to determine the recipients of a marketing campaign, the oracle dataset could easily contain thousands of instances.

GP has in many studies proved to be a very efficient search strategy. Often, GP results are comparable to, or even better than, results obtained by more specialized techniques. Specifically, several studies show that decision trees evolved using GP often are more accurate than trees induced by standard techniques like C4.5 and CART, see e.g., [11] and [12]. The main reason for this is that GP is a global optimization technique, while decision tree algorithms typically choose splits greedily, working from the root node down. Informally, this means that GP may make some locally sub-optimal splits, just as long as the overall model is more accurate.

III. Method

As mentioned in the introduction, the purpose of this study was to evaluate whether the use of a high-accuracy opaque model (serving as an oracle) may be beneficial for creating transparent predictive models. More specifically, we compared decision trees built from training data only to decision trees built using different combinations of training data and oracle data. The most important evaluation criterion was test accuracy, but we also compared training accuracies and fidelity towards the oracle. In this study, an ensemble of ANNs was used as the oracle. Details regarding the ANN ensembles used are given in the subsection *ANN ensembles* below. For the actual building of the decision trees, GP was used, i.e., all trees were evolved. The exact representation language used, GP parameters and other details for the evolution, are given in subsection *GP settings* below. For the experimentation, we used 4-fold cross-validation. On each fold, an ANN ensemble was first trained, using training data only. This ensemble (the oracle) was then applied to the test instances, producing test predictions. This gave us three different datasets:

- The ***training data*:** this is the original training dataset, i.e., original input vectors with corresponding correct target values.
- The ***ensemble data*:** this is the original training instances but with ensemble predictions as target values instead of the always correct target values.
- The ***oracle data*:** this is the test instances with corresponding ensemble predictions as target values.

In the experimentation, all different combinations of these datasets were evaluated as training data for the GP when evolving decision trees. In practice, this means that the GP fitness will reflect different combinations of training accuracy, training fidelity and test fidelity. More specifically, we had the following seven different setups:

- **Tree induction (I):** Standard tree induction using original training data only. This maximizes training accuracy.
- **Tree extraction (E):** Standard tree extraction, i.e., using ensemble data only. Maximizes training fidelity.
- **Prediction explanation (X):** Uses only oracle data, i.e., maximizes test fidelity.
- **Tree indanation**[1] **(IX):** Uses training data and oracle data, i.e., will maximize training accuracy and test fidelity.
- **Tree exduction (EI):** Uses training data and ensemble data. This means that if a specific training instance is misclassified by the ensemble, there will be two GP training instances having identical inputs but different target values. So, here training accuracy and training fidelity are simultaneously maximized.
- **Tree extanation (EX):** Uses ensemble data and oracle data, i.e., will maximize fidelity towards the ensemble on both training and test data.
- **Tree indextanation (IEX):** Uses all three datasets, i.e., will try to maximize training accuracy, training fidelity and test fidelity simultaneously.

Table I below summarizes the different setups.

TABLE I
SETUPS

Setup	Data used			Maximizes		
	Tr.	*Ens.*	*Oracle*	*Tr. acc.*	*Tr. Fid.*	*Test Fid.*
Induction (I)	X			X		
Extraction (E)		X			X	
Explanation (X)			X			X
Indanation (IX)	X		X	X		X
Exduction (EI)	X	X		X	X	
Extanation (EX)		X	X		X	X
Indextanation (IEX)	X	X	X	X	X	X

A. ANN ensembles

The opaque models used as oracles were ANN ensembles, each consisting of 15 independently trained ANNs. All

[1] These describing names, combining the terms *induction*, *extraction* and *explanation* in different ways, are of course made-up

ANNs were fully connected feed-forward networks where a localist representation was used. Averaging of posterior probabilities was used when determining ensemble classifications.

Of the 15 ANNs, seven had one hidden layer and the remaining eight had two hidden layers. The exact number of units in each hidden layer was slightly randomized to introduce some diversity, but used an heuristics based on number of inputs and classes in the current dataset. For ANNs with one hidden layer, the number of hidden units was determined from (1) below.

$$h = \left\lfloor 2 \cdot rand \cdot \sqrt{(v \cdot c)} \right\rfloor \quad (1)$$

Here, v is the number of input variables and c is the number of classes. *rand* is a random number in the interval [0, 1]. For ANNs with two hidden layers, the number of units in the first and second hidden layers were h_1 and h_2, respectively.

$$h_1 = \left\lfloor \sqrt{(v \cdot c)} / 2 + 4 \cdot rand \cdot \sqrt{(v \cdot c)} / c \right\rfloor \quad (2)$$

$$h_2 = \left\lfloor rand \cdot (\sqrt{(v \cdot c)} / c) + c \right\rfloor \quad (3)$$

B. GP settings

When using GP for rule (tree) extraction (induction), the available functions, *F*, and terminals *T*, constitute the literals of the representation language. Functions will typically be logical or relational operators, while the terminals could be, for instance, input variables or constants. Here, the representation language is very similar to basic decision trees. Figure 3 below shows a small but quite accurate (test accuracy is 0.771) sample tree evolved on the diabetes dataset.

```
if (Body_mass_index > 29.132)
 |T: if (plasma_glucose < 127.40)
 |   |T: [Negative] {56/12}
 |   |F: [Positive] {29/21}
 |F: [Negative] {63/11}
```

Figure 3: Sample tree evolved on diabetes dataset

The exact grammar used is presented using Backus-Naur form in Figure 4 below.

```
F = {if, ==, <, >}
T = {i1, i2, …, in, c1, c2, …, cm, ℜ}

DTree   :-  (if RExp Dtree Dtree) | Class
RExp    :-  (ROp ConI ConC) | (== CatI CatC)
ROp     :-  < | >
CatI    :-  Categorical input variable
ConI    :-  Continuous input variable
Class   :-  c1 | c2 | … | cm
CatC    :-  Categorical attribute value
ConC    :-  ℜ
```

Figure 4: Representation language used

The GP parameter settings used in this study are given in Table II below. The length penalty used is much smaller than the cost of misclassifying an instance. Nevertheless, the resulting parsimony pressure was able to significantly reduce the average program size in the population.

TABLE II
GP PARAMETER SETTINGS

Parameter	Value	Parameter	Value
Crossover rate	0.8	Creation depth	7
Mutation rate	0.01	Creation method	Ramped half-and-half
Population size	1500	Fitness function	Accuracy - length penalty
Generations	100	Selection	Roulette wheel
Persistence	25	Elitism	Yes

C. Datasets

The 26 datasets used are all publicly available from the UCI Repository [13]. For a summary of dataset characteristics, see Table III below. *Instances* is the total number of instances in the dataset. *Classes* is the number of output classes in the dataset. *Con.* is the number of continuous input variables and *Cat.* is the number of categorical input variables.

TABLE III
DATASET CHARACTERISTICS

Dataset	Instances	Classes	Con.	Cat.
Auto	205	7	15	10
Breast cancer (BC)	286	2	0	9
Colic	368	2	7	15
CMC	1473	3	2	7
Credit-A	690	2	6	9
Credit-G	1000	2	7	13
Diabetes	768	2	8	0
Glass	214	7	9	0
Haberman (Haber)	306	2	3	0
Heart-C	303	2	6	7
Heart-S	270	2	6	7
Hepatitis	155	2	6	13
Hypothyroid (Hypo)	3163	2	7	18
Iono	351	2	34	0
Iris	150	3	4	0
Labor	57	2	8	8
Liver	345	2	6	0
Sick	2800	2	7	22
Sonar	208	2	60	0
TAE	151	3	1	4
Tic-Tac-Toe (TTT)	958	2	0	9
Vehicle	846	4	18	0
Vote	435	2	0	16
Wine	178	3	13	0
Wisconsin breast cancer (WBC)	699	2	9	0
Zoo	100	7	0	16

IV. RESULTS

Table IV below shows training accuracies for all setups evaluated. Looking at the mean ranks, it is obvious that the three setups actually targeting training accuracy (I, EI and IEX) also obtain higher training accuracy.

TABLE IV
TRAINING ACCURACY

Dataset	I	E	X	IX	EI	EX	IEX
Auto	.672	.661	.484	.633	.696	.649	.675
BC	.769	.766	.745	.765	.783	.771	.779
Colic	.868	.866	.817	.869	.873	.867	.873
CMC	.562	.562	.557	.565	.563	.560	.559
Credit-A	.865	.867	.852	.867	.866	.865	.869
Credit-G	.741	.731	.713	.727	.726	.729	.725
Diabetes	.774	.771	.741	.778	.773	.764	.771
Glass	.702	.707	.590	.683	.717	.689	.705
Haber	.772	.773	.747	.773	.776	.778	.771
Heart-C	.861	.849	.781	.855	.859	.855	.852
Heart-S	.869	.861	.766	.846	.867	.849	.858
Hepatitis	.868	.885	.827	.882	.882	.865	.872
Hypo	.978	.974	.976	.979	.985	.981	.980
Iono	.935	.940	.860	.928	.935	.915	.938
Iris	.973	.962	.940	.973	.978	.962	.980
Labor	.983	.971	.791	.971	.988	.959	.977
Liver	.687	.670	.625	.700	.691	.690	.702
Sick	.978	.976	.960	.977	.964	.965	.974
Sonar	.811	.829	.670	.816	.833	.806	.840
TAE	.555	.537	.463	.537	.542	.546	.548
TTT	.806	.773	.728	.774	.801	.804	.823
Vehicle	.665	.676	.604	.668	.669	.654	.662
Vote	.966	.973	.937	.968	.967	.966	.969
Wine	.978	.963	.847	.968	.959	.970	.966
WBC	.970	.972	.944	.967	.971	.968	.973
Zoo	.924	.924	.865	.908	.908	.914	.924
Mean rank	**3.12**	**3.58**	**6.96**	**3.77**	**2.77**	**4.42**	**2.96**

TABLE V
TRAINING FIDELITY

Dataset	I	E	X	IX	EI	EX	IEX
Auto	.672	.661	.484	.633	.696	.649	.675
BC	.813	.813	.792	.809	.829	.817	.826
Colic	.870	.869	.820	.871	.876	.870	.876
CMC	.756	.790	.776	.761	.789	.788	.774
Credit-A	.909	.910	.897	.912	.912	.910	.912
Credit-G	.748	.738	.720	.733	.733	.736	.732
Diabetes	.864	.876	.841	.861	.872	.874	.869
Glass	.766	.776	.644	.744	.775	.756	.766
Haber	.929	.957	.900	.920	.955	.958	.943
Heart-C	.868	.856	.788	.863	.866	.863	.860
Heart-S	.871	.862	.767	.847	.868	.850	.860
Hepatitis	.868	.885	.827	.882	.882	.865	.872
Hypo	.984	.977	.981	.984	.988	.986	.986
Iono	.935	.940	.860	.928	.935	.915	.938
Iris	.973	.971	.940	.969	.982	.967	.985
Labor	.983	.971	.791	.971	.988	.959	.977
Liver	.736	.748	.680	.753	.756	.774	.764
Sick	.981	.984	.973	.981	.975	.977	.983
Sonar	.811	.829	.670	.816	.833	.806	.840
TAE	.588	.689	.542	.629	.654	.691	.664
TTT	.814	.795	.750	.795	.816	.824	.842
Vehicle	.675	.696	.611	.679	.685	.670	.673
Vote	.968	.975	.935	.969	.969	.968	.970
Wine	.978	.963	.847	.968	.959	.970	.966
WBC	.974	.976	.948	.971	.975	.971	.977
Zoo	.924	.924	.865	.908	.908	.914	.924
Mean rank	**3.77**	**2.96**	**6.85**	**4.46**	**2.73**	**4.04**	**2.81**

Although not the most important criterion, it should be noted that training accuracy of course represents how accurate the description of the relationship is, when considering the majority of the data. In situations requiring transparent models instead of just black-box prediction machines, this clearly has some value.

For a deeper analysis, we use the statistical tests recommended by Demšar [14] for comparing several techniques against each other over a number of datasets, i.e., a Friedman test [15], followed by a Nemenyi post-hoc test [16]. Evaluating seven setups using 26 datasets, the critical distance is as high as 1.76 (for α=0.05), i.e., the only statistically significant difference is that all other setups have higher training accuracy than X. Having said that, it is obvious that pair-wise comparisons, using for instance standard sign tests, would give a very different picture – a sign test would require 18 wins for statistical significance for α=0.05. One interesting comparison is between I and EI, where EI obtains higher training accuracy on a majority of the datasets (16 of 26 with one tie). This indicates that even just augmenting training data with ensemble data (i.e. without access to oracle data) may be successful. As a matter of fact, for training accuracy, EI has the best rank overall.

Table V below shows training fidelities for all setups evaluated. Here, too, the results are as expected, since E, EI and IEX obtain the highest fidelities on training data.

Table VI below shows test fidelities. Again, setups actually targeting test fidelity (X, IX and IEX) obtain the best results.

TABLE VI
TEST FIDELITY

Dataset	I	E	X	IX	EI	EX	IEX
Auto	.627	.618	.765	.730	.676	.716	.662
BC	.835	.842	.884	.870	.856	.877	.849
Colic	.899	.910	.943	.924	.897	.916	.905
CMC	.762	.800	.827	.791	.798	.787	.783
Credit-A	.933	.920	.942	.942	.919	.940	.939
Credit-G	.832	.840	.855	.859	.813	.856	.843
Diabetes	.858	.852	.891	.870	.850	.889	.865
Glass	.741	.745	.901	.797	.778	.769	.807
Haber	.928	.918	.987	.977	.918	.974	.964
Heart-C	.853	.817	.923	.883	.820	.900	.870
Heart-S	.821	.843	.910	.862	.828	.858	.851
Hepatitis	.868	.875	.954	.947	.882	.934	.928
Hypo	.985	.979	.984	.987	.987	.987	.987
Iono	.928	.928	.960	.934	.908	.925	.917
Iris	.932	.959	.993	.980	.966	.973	.966
Labor	.857	.893	1.00	.964	.893	1.00	.964
Liver	.712	.735	.828	.791	.718	.756	.727
Sick	.970	.977	.974	.978	.970	.975	.979
Sonar	.736	.764	.933	.813	.760	.827	.750
TAE	.581	.595	.736	.682	.574	.676	.655
TTT	.829	.831	.865	.872	.846	.889	.886
Vehicle	.684	.682	.714	.719	.682	.688	.679
Vote	.965	.977	.988	.979	.972	.979	.979
Wine	.926	.898	.989	.966	.920	.943	.949
WBC	.970	.973	.989	.983	.964	.983	.977
Zoo	.900	.920	.970	.970	.920	.960	.940
Mean rank	**6.04**	**5.23**	**1.58**	**2.23**	**5.58**	**2.88**	**4.15**

One interesting observation is that combining training data with oracle data (IX) actually produced higher test fidelity than combining ensemble data and oracle data (EX) on a large majority of the datasets (18 wins of 26 with two ties).

Table VII below shows the results for the most important criterion, i.e., test accuracy.

TABLE VII
TEST ACCURACY

Dataset	**Ens.**	**I**	**E**	**X**	**IX**	**EI**	**EX**	**IEX**
Auto	.735	.613	.603	.642	.618	.647	.637	.647
BC	.715	.746	.739	.754	.711	.732	.739	.746
Colic	.832	.845	.845	.840	.853	.853	.851	.834
CMC	.547	.550	.554	.545	.548	.547	.539	.558
Credit-A	.866	.846	.859	.849	.858	.849	.850	.846
Credit-G	.763	.723	.707	.730	.728	.706	.727	.728
Diabetes	.755	.738	.747	.750	.742	.751	.770	.737
Glass	.689	.656	.660	.689	.670	.670	.627	.665
Haber	.737	.730	.753	.737	.740	.760	.757	.747
Heart-C	.817	.803	.773	.827	.807	.797	.777	.807
Heart-S	.787	.780	.817	.802	.813	.802	.817	.825
Hepatitis	.849	.822	.816	.855	.875	.822	.862	.842
Hypo	.984	.978	.976	.976	.980	.982	.980	.980
Iono	.931	.917	.922	.920	.911	.902	.908	.905
Iris	.973	.932	.959	.966	.980	.966	.959	.966
Labor	.946	.875	.911	.946	.911	.875	.946	.946
Liver	.709	.631	.654	.672	.692	.608	.657	.674
Sick	.968	.978	.974	.957	.973	.964	.960	.974
Sonar	.861	.750	.760	.832	.740	.736	.793	.736
TAE	.547	.541	.527	.547	.541	.493	.507	.527
TTT	.881	.788	.757	.752	.768	.798	.799	.823
Vehicle	.846	.656	.658	.673	.681	.659	.671	.648
Vote	.963	.951	.968	.961	.965	.954	.961	.965
Wine	.977	.932	.909	.966	.966	.926	.943	.949
WBC	.963	.955	.958	.966	.957	.953	.963	.954
Zoo	.960	.910	.920	.980	.950	.930	.960	.960
Mean rank	**N/A**	**4.96**	**4.27**	**3.04**	**3.15**	**4.46**	**3.62**	**3.42**

The key observation is of course that standard tree induction (I) obtains the worst test set accuracy of all setups evaluated. Comparing I to all other setups using another Friedman test, but now followed by a Bonferroni-Dunn post-hoc test, since we compare one "control" technique against all others, the critical distance becomes 1.58. So, both X and IX obtained significantly higher test set accuracy than I in this experiment. In addition, IEX is very close to also being significantly more accurate than I.

Table VIII below shows pair-wise comparisons (wins, ties and losses for the row setup against the column setup) between the seven different setups. Statistically significant differences (based on sign tests) are shown using bold and underlined values.

TABLE VIII
TEST ACCURACY – WINS, TIES AND LOSSES.

	I	E	X	IX	EI	EX	IEX
I	-	10-1-15	**5-0-21**	**6-1-19**	11-2-13	**7-0-19**	8-2-16
E	15-1-10	-	9-1-16	11-1-14	13-0-13	7-3-16	9-2-15
X	**21-0-5**	16-1-9	-	12-1-13	15-3-8	15-2-9	15-2-9
IX	**19-1-6**	14-1-11	13-1-12	-	**18-2-6**	15-1-10	13-4-9
EI	13-2-11	13-0-13	8-3-15	**6-2-18**	-	9-0-17	7-3-16
EX	**19-0-7**	16-3-7	9-2-15	10-1-15	17-0-9	-	9-3-14
IEX	16-2-8	15-2-9	9-2-15	9-4-13	16-3-7	14-3-9	-

Table IX below, which specifically compare all setups not using oracle data, also contain results for the J48 algorithm from the Weka workbench [17]. In this experiment, J48 which is an implementation of the C4.5 algorithm [1], was run with default settings.

TABLE IX
COMPARISON OF TECHNIQUES NOT USING ORACLE DATA (TEST ACCURACY)

Dataset	**I**	**E**	**EI**	**J48**
Auto	.613	.603	.647	.763
BC	.746	.739	.732	.721
Colic	.845	.845	.853	.851
CMC	.550	.554	.547	.508
Credit-A	.846	.859	.849	.854
Credit-G	.723	.707	.706	.718
Diabetes	.738	.747	.751	.743
Glass	.656	.660	.670	.676
Haber	.730	.753	.760	.707
Heart-C	.803	.773	.797	.765
Heart-S	.780	.817	.802	.782
Hepatitis	.822	.816	.822	.794
Hypo	.978	.976	.982	.996
Iono	.917	.922	.902	.894
Iris	.932	.959	.966	.938
Labor	.875	.911	.875	.766
Liver	.631	.654	.608	.632
Sick	.978	.974	.964	.987
Sonar	.750	.760	.736	.714
TAE	.541	.527	.493	.548
TTT	.788	.757	.798	.839
Vehicle	.656	.658	.659	.721
Vote	.951	.968	.954	.962
Wine	.932	.909	.926	.932
WBC	.955	.958	.953	.949
Zoo	.910	.920	.930	.933
Mean rank	**2.69**	**2.31**	**2.42**	**2.46**

As indicated by the mean ranks in Table IX, both setups using ensemble data (E and EI) outperformed J48 and the tree induction (I), even if the differences are far from significant. It is of course interesting to observe that the extracted model (E) turns out to be more accurate than both the induced (I) and J48. This must be considered a strong argument for rule extraction in general.

Turning to setups using oracle data, the best rank, in Table VII, is achieved by using oracle data only (X). Furthermore, on a majority of the datasets, the test accuracy obtained using X is actually better than or similar to the ensemble. This, together with the very high test fidelity achieved, is of course a very encouraging result. So, if the purpose is to explain or understand the basis for predictions made, the best choice is probably to use oracle data only, i.e., the X setup. At the same time, it should be noted that X had the worst training accuracy and training fidelity, so it is not a very good description of the original training data. So, to get a slightly more balanced description, using training data and oracle data (IX), appears to be a very good choice. Although training accuracy and fidelity are rather poor, they are still significantly better than for X. Test set fidelity and accuracy are on the other hand excellent. As a matter of fact, IX actually outperformed all other setups (including X) when counting wins and losses for test set accuracy.

V. Conclusions

We have in this paper argued for using an oracle guide when producing predictive models required to be comprehensible. The suggested technique uses a high-accuracy predictive model, here an ANN ensemble, to produce ensemble or oracle data, i.e., training or test data instances, together with corresponding predictions from the opaque model. These instances, together with the original training data, could then, in different combinations, be used as training data by another data mining technique when building transparent models. In this study, GP was used to evolve decision trees, and altogether seven different ways of combining training, ensemble and oracle data were evaluated.

From the results, obtained using 26 UCI datasets, it is obvious that the use of especially oracle, but also ensemble data, led to an increase in test set accuracy. Trees evolved using training data only in fact had the worst test set accuracy of all seven setups evaluated. Two of the evaluated setups, using either only oracle instances, or oracle instances together with original training instances, actually produced significantly more accurate trees, compared to the setup using training data only. So, since the transparent models evolved using ensemble or oracle data had higher accuracy on the test set; these models explain their predictions made on the novel data better than the trees evolved using training data only.

VI. Discussion and future work

First of all, it is very important to recognize the situation targeted in this paper, i.e., that for some reason a black-box prediction machine is not sufficient. If comprehensibility is not an issue, there is no reason to use techniques like decision trees or rule sets, since these will almost always be outperformed by neural networks or ensembles. Having said that, it should be noted that it is neither "cheating" nor very complicated to use oracle data when building predictive models. On problems where predictions are made for sets of instances, it is actually a fairly straightforward process.

In the targeted situation, test set accuracy is still the most important criterion, but other criteria like training accuracy and test fidelity are also indicators of the model quality. Using oracle data, the most straightforward interpretation of higher test set accuracy, is that it constitutes a better explanation of the predictions made.

In this study, we used GP to evolve all decision trees utilizing oracle data. The suggested approach is, however, also applicable to standard algorithms like C4.5 and CART. Evaluating the use of oracle data for algorithms constructing decision trees greedily is a prioritized future study.

During experimentation, we used only training data and test data. Often, training data is split in two parts, where one part (the validation set) is used to somehow select a specific trained model to apply to the test data. Here, separate parts of the training data could be used for training the ensemble and for the evolution, thus encouraging more general models. Investigating whether the use of a validation set would improve the performance or not could be the focus of a future study. Naturally, using fresh data for the evolution would introduce even more possible combinations of different datasets, so exactly how the dataset should be used optimally must also be addressed in such a study.

Finally, the use of multi-objective fitness functions should be evaluated as an alternative to combining several properties (e.g. training accuracy and test fidelity) into one fitness function.

Acknowledgment

This work was supported by the Information Fusion Research Program (University of Skövde, Sweden) in partnership with the Swedish Knowledge Foundation under grant 2003/0104 (URL: http://www.infofusion.se).

References

[1] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[2] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, Wadsworth International, 1984.

[3] R. Andrews, J. Diederich and A. B. Tickle, A survey and critique of techniques for extracting rules from trained artificial neural networks, *Knowledge-Based Systems*, 8(6), 1995.

[4] U. Johansson, T. Löfström, R. König and L. Niklasson, Why Not Use an Oracle When You Got One?, *Neural Information Processing - Letters and Reviews*, Vol. 10, No. 8-9: 227-236, 2006.

[5] H. Lu, R. Setino and H. Liu, Neurorule: A connectionist approach to data mining, *International Very Large Databases Conference*, pp. 478-489, 1995.

[6] M. Craven and J. Shavlik, Extracting Tree-Structured Representations of Trained Networks, *Advances in Neural Information Processing Systems*, 8:24-30, 1996.

[7] M. Craven and J. Shavlik, Rule Extraction: Where Do We Go from Here?, *University of Wisconsin Machine Learning Research Group working Paper*, 99-1, 1999.

[8] U. Johansson, R. König and L. Niklasson, Rule Extraction from Trained Neural Networks using Genetic Programming, *13th International Conference on Artificial Neural Networks*, Istanbul, Turkey, supplementary proceedings pp. 13-16, 2003.

[9] R. König, U. Johansson and L. Niklasson, G-REX: A Versatile Framework for Evolutionary Data Mining, *IEEE International Conference on Data Mining (ICDM08)*, Pisa, Italy, Demo paper, Workshop Proceedings, pp. 971-974, 2008.

[10] U. Johansson, *Obtaining accurate and comprehensible data mining models: An evolutionary approach*, PhD thesis, Institute of Technology, Linköping University, 2007.

[11] A. Tsakonas, A comparison of classification accuracy of four genetic programming-evolved intelligent structures, *Information Sciences*, 176(6):691-724, 2006.

[12] C. C. Bojarczuk, H. S. Lopes and A. A. Freitas, Data Mining with Constrained-syntax Genetic Programming: Applications in Medical Data Sets, *Intelligent Data Analysis in Medicine and Pharmacology - a workshop at MedInfo-2001*, 2001.

[13] A. Asuncion and D. J. Newman, *UCI machine learning repository*, 2007.

[14] J. Demšar, Statistical Comparisons of Classifiers over Multiple Data Sets, *Journal of Machine Learning Research*, 7:1–30, 2006.

[15] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, *Journal of American Statistical Association*, 32:675–701,1937.

[16] P. B. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.

[17] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd edition, Morgan Kaufmann, 2005.