Accuracy of Machine Learning Models versus "Hand Crafted" Expert Systems - A Credit Scoring Case Study

Arie Ben-David
Management Information Systems
Dept. of Technology Management
Holon Institute of Technology (*)
Holon, Israel

Eibe Frank
Dept. of Computer Science
University of Waikato
Hamilton
New Zealand

Abstract

Relatively few publications compare machine learning models with expert systems when applied to the same problem domain. Most publications emphasize those cases where the former beat the latter. Is it a realistic picture of the state of the art?

Some other findings are presented here. The accuracy of a real world "mind crafted" credit scoring expert system is compared with dozens of machine learning models. The results show that while some machine learning models can surpass the expert system's accuracy with statistical significance, most models do not. More interestingly, this happened only when the problem was treated as regression. In contrast, no machine learning model showed any statistically significant advantage over the expert system's accuracy when the same problem was treated as classification. Since the true nature of the class data was ordinal, the latter is the more appropriate setting. It is also shown that the answer to the question is highly dependent on the meter that is being used to define accuracy.

Keywords : **Machine Learning Models, Expert Systems, Accuracy, Classification, Regression, Hit Ratio, Cohen's Kappa, Credit Scoring.**

(*) **Correspondence Address: Dr. Arie Ben-David, MIS, Dept. of Technology Management, Holon Institute of Technology, 52 Golomb St. P.O.Box 305 Holon 58102 , Israel. Phone: (972) 3 5026744/6 Fax: (972) 3 5026650 E.Mail: hol_abendav@bezeqint.net.il**

1. Introduction

Emerging technologies are often accompanied by some degree of hype. During the Eighties there has been a flood of success stories on how expert systems out-performed human experts. Similar reports were published during the Nineties, when machine learning models became increasingly popular. Very few research publications, however, have compared the capabilities of these two approaches: expert systems and machine learning, when applied to similar problems. Furthermore, as expert systems began to include machine learning components, the boundary between these two approaches began to blur, making a direct comparison between expert systems and machine learning quite difficult.

The term expert system is used here to describe a computerized system that encapsulates human knowledge, without resorting to any machine learning or data mining technique. Such systems are typically built using a process nicknamed "knowledge engineering", in which human experts are being interviewed. The end product of this process is a computer program that tries to mimic the way the experts solve the particular problem, usually in some sort of rule-base form. Machine learning models, on the other hand, usually need only data about past decisions. Given this data, they are supposed to generate a model that effectively solves the problem without any further human assistance.

It is not an easy task to compare expert systems and machine learning. Good commercial machine learning and expert systems are often kept secret, due to fear of competition. Unsuccessful implementations rarely find their way to the literature for obvious reasons. Furthermore, most organizations are not willing to commit sufficient resources for developing both types of implementations for solving the same problem. For these reasons, most publications compare either expert systems or machine learning with human performance. Publications that compare expert systems and machine learning, when applied to the same problem domain, are very rare.

Why is it of interest to compare expert systems with machine learning? Besides academic curiosity, an answer to this question is of importance to any commercial firm and government organization that invests or considers investing money in either or both technologies. While there are many pros and cons of each approach, here we only deal with the accuracy aspect through a test case. The other considerations, though very important, are outside the scope of this research.

This research, thus, does not aim at providing the ultimate answer to the question whether machine learning is preferable to expert systems or the other way around. We were fortunate, though, and had a commercial credit scoring expert system handy with some data that enabled us to test dozens of state-of-the-art machine learning models and compare their accuracy with that of the expert system. To the best of our knowledge this is the first time such a comprehensive comparison study is published in the literature.

2. Related Work

Comparing various aspects of machine learning models and expert systems has long been of interest to researchers from various disciplines such as machine learning, artificial intelligence, and decision-making - in particular, how accurate machine learning models are when compared with expert systems. However, there have been very few reports that directly compare the two approaches when applied to the same problem. Most publications compare either expert systems or machine learning with human expert performance.

One of the earliest comparisons between machine learning and expert systems can be found in the landmark paper entitled "Knowledge acquisition by encoding expert rules versus computer induction from examples: A case study of involving soybean pathology" by Michalski and Chilausky (1980). They found that in that particular case, a machine learning model was more accurate than the expert system.

In the area of medical diagnosis, Musen (1989) reported an advantage of an expert system over traditional statistical methods, now being in use in machine learning.

Another well-known comparison was published by Creecy et al. (1992). It reports a clear accuracy-wise advantage of a version of the k-nearest neighbor algorithm over an expert system that was in use at the US Census Bureau.

A recent publication published by Darren et al. (2005) also reports an advantage of a machine learning model, a feed forward neural network, over an expert system in terms of accuracy, in the domain of quality control.

Unlike the above reports, which compare the accuracy of a single expert system with a one machine learning model, this research performs a comparison with dozens of such models, making it the most comprehensive, methodological, research published in the literature to date. The findings of this research are not all confirmatory with previous reports. On the contrary, as will shortly be seen, unlike the above reports that claim accuracy-wise advantage of machine learning models over expert systems, no statistically significant clear-cut winner was found in the classification experiment to be described shortly.

3. The Experiment

The data set contained 390 examples of consumer loan credit scoring decisions, made by a real-world expert system in one of Israel's leading financial institutions. Both the expert system's decisions and the correct output were available. The input variables were mixed: some numeric, some ordinal, and some were nominal. The output, was ordinal, represented by numbers. It could have been symbolic as well though (i.e., "excellent", "very good", "good", etc.).

The Weka machine learning workbench (Witten and Frank, 2005), which has become a standard benchmarking tool in the machine learning community in recent years, was used throughout this experiment. Version 3.4.7 of Weka was used. All the results are based on ten-fold cross-validation. Unlike machine learning models, which are re-built from scratch with every fold, the expert system was not rebuilt each time, since it was a

unified, complete structure. However, while the expert system remained fixed, the testing data that was fed into it was identical to the data used to validate the accuracy of the machine learning models. At each fold the expert system was fed with one tenth of the testing data, and both the predicted and the true values were recorded. All the machine learning algorithms were used with their default parameter settings, as defined in Weka 3.4.7, to reduce the danger of overfitting due to excessive parameter tuning.

Since only one model out of the dozens currently embedded in Weka uses the order within ordinal class values (OrdinalClassClassifier), and due to the fact that ordinal scales share some numeric properties (i.e., the order) as well as some nominal features (i.e., lack of distance), it was decided to conduct two experiments. A: As a classification problem. B. As a regression one. This way, the number of machine learning models that could be used in the experiment increased significantly, as models that can do both regression and classification are relatively rare. Stratified ten fold cross-validation was used for the nominal-class version of the experiment, while non-stratified ten fold cross-validation was used in the regression version. The number of possible class values in the classification experiment was ten.

The major findings of the two experiments are given in the following section. A short description of the models which were used, as well as comments about their Weka implementation and their relative accuracy follow in the Discussion section.

4. Major Findings

The results of the classification experiment are presented first in this section, followed by those of regression.

Most classification-related publications in the machine learning literature consider the hit ratio as the major meter for accuracy. Table 1, therefore, shows the classification experiment results sorted by decreasing order of the average hit ratio. The rank of the model, its name in Weka, the percentage of correct classifications (i.e., the hit ratio), half the width of a 95% confidence interval, and the group to which the model belongs in

Weka (usually reflecting the type of concept being learned, e.g., mathematical function, decision tree, etc.), are shown from left to right. The expert system's hit ratio is ranked in the 21st position of Table 1, and it is written in bold letters.

Rank	***Model***	***HIT %***	***HIT 95% half width CI***	***Group***
1	NaiveBayes	48.72	5.469	Bayes
2	ConjunctiveRule	47.69	3.789	Rules
3	LWL	47.69	3.586	Lazy
4	AdaboostM1	47.69	3.586	Meta
5	MultiboostAB	47.69	3.586	Meta
6	DecisionStump	47.69	3.586	Trees
7	RBFNetwork	47.44	5.060	Functions
8	OneR	46.92	3.674	Rules
9	SimpleLogistic	46.67	5.455	Functions
10	LMT	46.67	5.455	Trees
11	RepTree	46.41	4.000	Trees
12	BayesNet	46.16	6.642	Bayes
13	LogitBoost	45.64	5.174	Meta
14	DecisionTable	45.64	6.459	Rules
15	SMO	45.38	5.119	Functions
16	Bagging	45.13	4.830	Meta
17	ClassificationViaRegression	44.62	2.476	Meta
18	RandomForest	44.10	4.876	Trees
19	AttributeSelectedClassifier	44.10	3.648	Meta
20	OrdinalClassClassifier	43.85	4.926	Meta
21	**EXPERT SYSTEM**	**43.33**	**5.763**	**------**
22	Decorate	43.08	3.648	Meta
23	MulltilayerPerceptron	42.31	5.752	Functions
24	NBTree	42.05	5.941	Trees
25	NNGE	41.54	6.574	Rules
26	RandomCommittee	41.28	4.849	Meta
27	FilteredClassifier	40.51	4.952	Meta
28	MulticlassClassifier	40.26	5.334	Meta
29	Logistic	39.74	5.134	Functions
30	J48	39.49	4.752	Trees
31	Ridor	38.97	5.723	Rules
32	Part	37.44	2.623	Rules
33	Kstar	36.67	3.774	Lazy
34	IBK	35.38	4.306	Lazy
35	JRIP	34.36	3.789	Rules
36	RandomTree	33.85	5.101	Trees
37	ZeroR	29.23	0.947	Rules

Table 1 - Classification Accuracy Sorted by Hit Ratio

It is not surprising to see in Table 1 (and later in Tables 2 and 3) that given a dataset, machine learning models vary significantly from each other in terms of their accuracy (see Lim et al., 2000, for instance). However, unlike other reports, no machine learning model which was tested here had any statistically significant advantage over the expert system in terms of hit ratio. This can clearly be seen Figure 1, which shows a 95% confidence interval around the average hit ratios of all the tested models. The distance between the mean hit ratio and the symmetric boundaries of this two-sided 95% confidence interval is labeled " 95% half width CI" in Table 1.

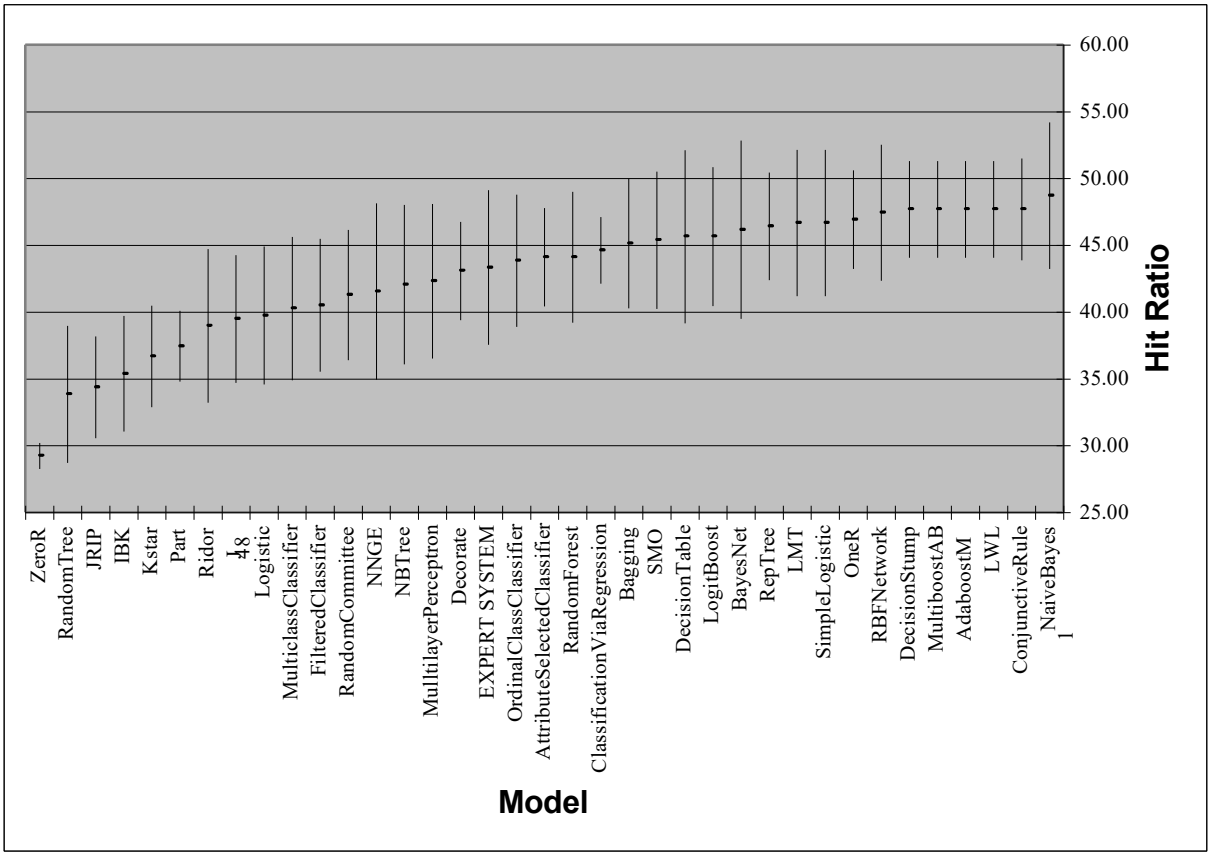

Figure 1: Hit Ratio - 95% Confidence Intervals

A one-sided paired t-test of each of the twenty machine learning models that ranked before the expert system in terms of hit ratio against the expert system has also been carried out. It also failed to reveal any statistically significant advantage ($\alpha = 0.05$) of any machine learning model over the expert system.

The hit ratio meter ignores classifications that might have been due to mere chance. Cohen's Kappa statistic, on the other hand, compensates for that (Cohen, 1960). For this reason, Cohen's Kappa is considered a more accurate meter for measuring the accuracy of both binary and multi-class classification problems (Cicchetti, 1990; Cook, 1998). It is not widely used yet, though, as a primary meter for classification accuracy in the machine learning community. However, due to its theoretical merits and the fact that Cohen's Kappa has widely been used in other disciplines such as Statistics and Medicine for decades, it was decided not to overlook it in this experiment. Table 2, which is similar in structure to Table 1, shows that only two machine learning models, namely Naive Bayes and RBF Networks out-ranked the expert system in terms of Kappa, while thirty-five models lagged behind. This is an unexpected results.

More importantly, based on both two sided 95% confidence intervals, shown graphically in Figure 2, and one sided hypothesis tests ($\alpha = 0.05$), the advantage of both Naive Bayes and RBF Networks over the expert system in terms of Kappa was statistically insignificant. These were not the results one would expect when reading the publications mentioned earlier in the Related Work section, which reported an advantage of machine learning models over expert systems.

Rank	*Model*	*KAPPA*	*KAPPA 95% half width CI*	*Group*
1	NaiveBayes	0.3292	0.0727	Bayes
2	RBFNetwork	0.3148	0.0676	Functions
3	**EXPERT SYSTEM**	**0.3116**	**0.0778**	**------**
4	BayesNet	0.3016	0.0841	Bayes
5	LogitBoost	0.2887	0.0670	Meta
6	SMO	0.2869	0.0644	Functions
7	DecisionTable	0.2765	0.0837	Rules
8	RepTree	0.2711	0.0554	Trees
9	SimpleLogistic	0.2685	0.0626	Functions
10	LMT	0.2685	0.0626	Trees
11	ConjunctiveRule	0.2665	0.0499	Rules
12	LWL	0.2664	0.0472	Lazy
13	AdaboostM1	0.2664	0.0472	Meta
14	MultiboostAB	0.2664	0.0472	Meta
15	DecisionStump	0.2664	0.0472	Trees
16	Bagging	0.2664	0.0572	Meta
17	RandomForest	0.2648	0.0639	Trees
18	Decorate	0.2622	0.0510	Meta
19	AttributeSelectedClassifier	0.2619	0.0496	Meta
20	OrdinalClassClassifier	0.2604	0.0624	Meta
21	ClassificationViaRegression	0.2596	0.0328	Meta
22	OneR	0.2568	0.0477	Rules
23	Logistic	0.2491	0.0610	Functions
24	MulticlassClassifier	0.2483	0.0645	Meta
25	NNGE	0.2473	0.0797	Rules
26	MulltilayerPerceptron	0.2465	0.0753	Functions
27	NBTree	0.2332	0.0810	Trees
28	Ridor	0.2251	0.0712	Rules
29	J48	0.2174	0.0556	Trees
30	FilteredClassifier	0.2128	0.0603	Meta
31	RandomCommittee	0.2097	0.0652	Meta
32	Part	0.1959	0.0339	Rules
33	Kstar	0.1717	0.05	Lazy
34	IBK	0.1651	0.0548	Lazy
35	RandomTree	0.1538	0.0690	Trees
36	JRIP	0.1029	0.0553	Rules
37	ZeroR	0.0000	0.0000	Rules

Table 2 - Classification Accuracy Sorted by Cohen's Kappa

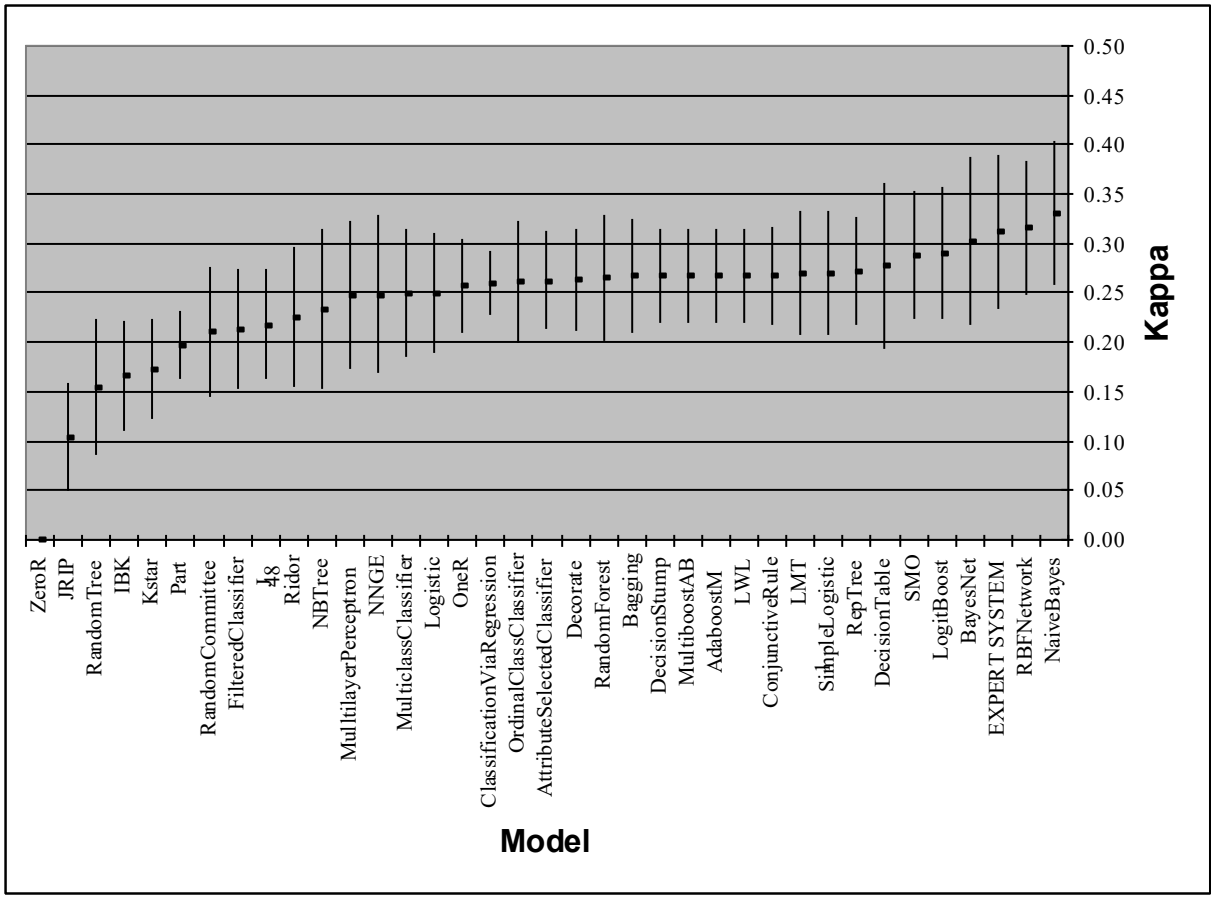

Figure 2: Kappa - 95% Confidence Intervals

Table 3 shows the results of the regression experiment. The structure of Table 3 is similar to that of Tables 1 and 2. The models are ranked by the Mean Absolute Error (MAE), which is a common accuracy meter in the literature. Figure 3 shows a 95% confidence interval of the MAE.

Rank	*Model*	*MAE*	*95% half width CI*	*Group*
1	LeastMedSq	0.0716	0.0103	Functions
2	M5Rules	0.0748	0.0073	Rules
3	M5P	0.0751	0.0068	Trees
4	SMOReg	0.0763	0.0093	Functions
5	Bagging	0.0778	0.0116	Meta
6	LineaRegression	0.0787	0.0074	Functions
7	DecisionTable	0.0788	0.0115	Rules
8	LWL	0.0814	0.0110	Lazy
9	RegressionByDescretization	0.0828	0.0109	Meta
10	REPTree	0.0828	0.0108	Trees
11	ConjuctiveRule	0.0833	0.0110	Rules
12	AdditiveRegression	0.0855	0.0105	Meta
13	DecisionStump	0.0878	0.0095	Trees
14	**EXPERT SYSTEM**	**0.0900**	**0.0055**	**--------**
15	RBFNetwork	0.0909	0.0135	Functions
16	Kstar	0.0930	0.0175	Lazy
18	ZeroR	0.1069	0.0104	Rules
19	IBK	0.1070	0.0113	Lazy
20	MultilayerPerceptron	0.1232	0.0151	Functions

Table 3 - Regression Accuracy Sorted by Mean Absolute Error

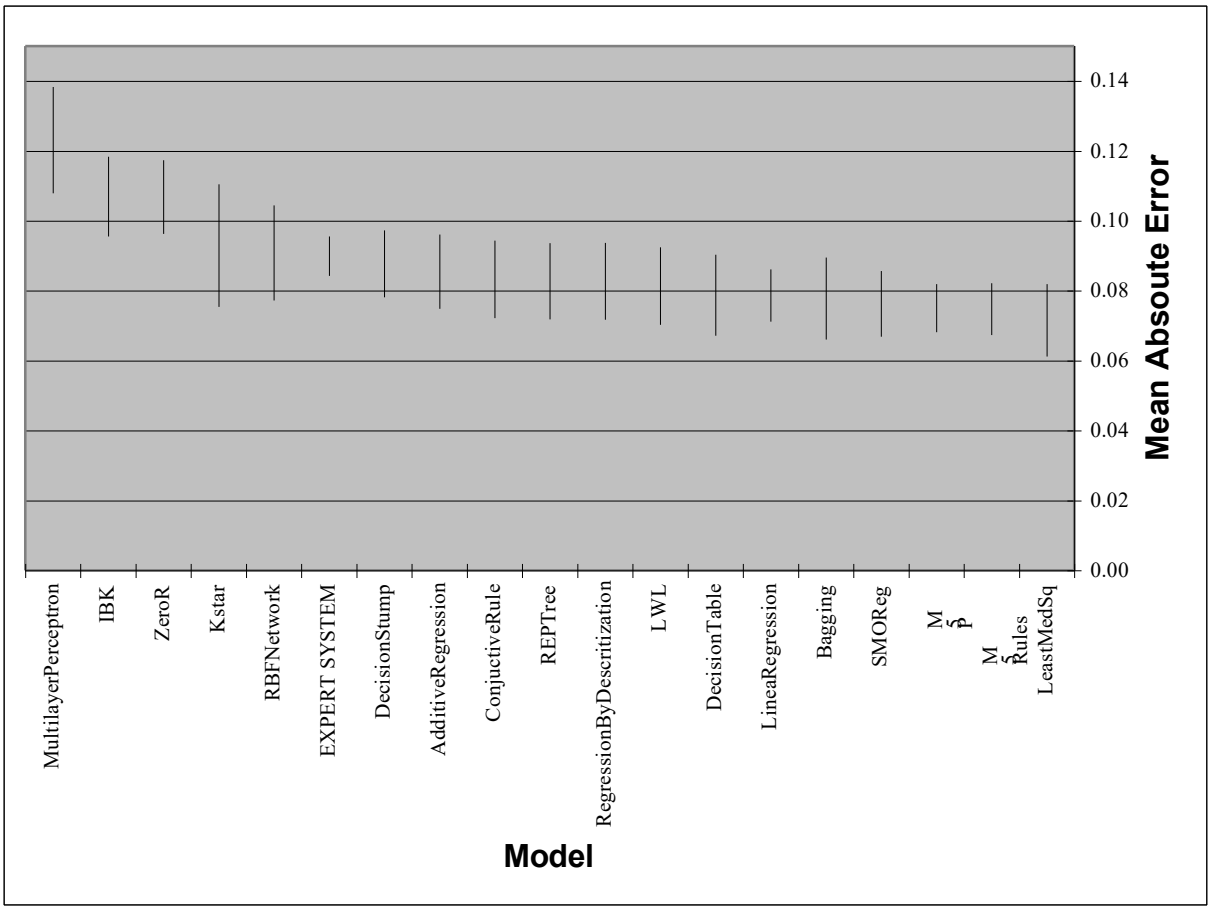

Figure 3: Mean Absolute Error - 95% Confidence Intervals

The results of the regression experiment shown in Table 3 and Figure 3 are quite different from those described above for classification. Here the expert system ranked fourteenth out of 20 models. According to the two-sided 95% confidence interval of Figure 3, only LeastMedSq, M5Rules and M5P have a statistically significant advantage over the expert system in terms of MAE. However, the results of a one-sided hypothesis test show that the above three models, as well as SMOReg, Bagging, Linear Regression and Decision Tables had a statistically significant advantage over the expert system at $\alpha = 0.05$. The findings of the regression experiment are more in line with the reports in the literature discussed earlier, while those of the classification experiment are not.

5. Discussion

The results of this experiment were quite unexpected. Based on the publications mentioned above, it was expected that many machine learning models will exhibit a significant statistical advantage over the expert system's rules, which were manually "mind-crafted" by human experts. Nevertheless, it turned out that no machine learning model had any statistically significant advantage over the expert system in terms of hit ratio and Kappa statistic in the classification experiment, and seven had such an advantage in the regression case.

Our results indicate that a naïve user of machine learning, who is not schooled in the intricacies of the machine learning methods that are at his or her disposal, will struggle to beat the performance of a carefully hand-crafted expert system.

In the classification experiment, Naive Bayes ranked first. Naive Bayes builds a simple probabilistic model based on the assumption that all attributes are conditionally independent given the class attribute. The implementation in Weka assumes that numeric attributes are normally distributed and uses the Laplace correction to eliminate the zero-frequency problem in the nominal case (Witten and Frank, 2005). It is well-known that naive Bayes can perform very well on classification problems even if the independence assumption does not hold (Domingos and Pazzani, 1997).

Locally weighted learning (Atkenson et al, 1997) is a lazy learning technique that makes a prediction by taking a weighted neighborhood of the test instance, building a classifier using this weighted subset, and obtaining a prediction from this classifier. The default implementation in Weka uses a linear weighting function that assigns weights to all instances in the data and builds a one-level decision tree (Witten and Frank, 2005) based on the weighted data. It appears that locally weighted learning actually adds little to just building a decision stump from the original, unweighted data, as can be seen of the relative performance of LWL and DecisionStump in Tables 1 and 2.

Decision Stump only uses one attribute in the data for prediction. This is also the case for the OneR scheme (Holte, 1993), which differs from DecisionStump only in that it uses accuracy instead of information gain to select an attribute and generates a multi-way split instead of a binary split on numeric attributes. Similar to other reports (Holte, 1993), these simple algorithms performed fairly well on this data set.

The boosting schemes AdaBoostM1 (Freund and Schapire, 1996) and MultiBoostAB (Webb, 2000) also perform similarly, and, as the Weka implementations of these ensemble learning schemes use decision stumps as the base models by default, this result means that boosting was unsuccessful in improving performance. ConjunctiveRule builds a single if-then rule from the data, and uses the majority class as the default prediction for instances not covered by the rule, and also performed relatively well. The accuracy achieved by these models relative to more complex models show that the data set included very few attributes that actually affect the outcome.

Good relative performance, both in terms of hit ratio and Kappa statistic was achieved by the Weka implementation of a normalized Gaussian radial basis function network (Bishop, 1995). Given this result, and the good performance of naive Bayes, we conclude that the numeric attributes in this data were modeled well using per-class normal distributions

SimpleLogistic (Landwehr et al., 2005) implements a logistic regression model with cross-validation-based attribute selection and performs well, in comparison with the full maximum-likelihood logistic regression model that is produced by Weka's Logistic class. LMT builds a logistic model tree (Landwehr et al, 2005), a decision tree with logistic models at the leaf nodes. The fact that the performance of LMT and SimpleLogistic is the same means LMT actually builds a degenerate tree that consists of a single node, as SimpleLogistic is used to build the logistic models for the tree.

Another, much simpler, tree learner in Weka is REPTree, which uses reduced-error pruning (Quinlan, 1987) to avoid overfitting, and the information gain criterion to select splits at the internal nodes. It is quite surprising that it performed better than J48 on this

data. The latter is Weka's implementation of the C4.5 decision tree learner (Quinlan, 1993).

BayesNet is Weka's implementation of a Bayesian network learner (Witten and Frank, 2005). In default mode it builds a naive Bayes classifier. However, in contrast to NaiveBayes, numeric attributes are discretized using supervised MDL-based discretization (Fayyad and Irani, 1993).

The LogitBoost boosting algorithm (Friedman et al., 2000) was ranked somewhat worse than the AdaboostM1 boosting algorithm in terms of accuracy, but it ranked better in terms of the Kappa statistic. LogitBoost implements additive logistic regression, whereas AdaBoostM1 optimizes an exponential loss function using an additive model (Friedman et al., 2000). Like the default version of AdaBoostM1 in Weka, LogitBoost uses decision stumps as the base learner.

DecisionTable builds a simple decision table using best-first search to find a good subset of attributes (Kohavi, 2005), and SMO implements the sequential minimal optimization algorithm (Platt, 1998) for learning a linear support vector machine (in default mode). Bagging (Breiman, 1996) builds an ensemble of ten decision trees from sub-sampled version of the data using REPTree as the base learner. ClassificationViaRegression uses model trees (Quinlan, 1992), regression trees with linear regression functions at the leaves, for classification, by creating one regression problem with an indicator variable for each class. AttributeSelectedClassifier combines the J48 decision tree learner with a correlation-based filter for attribute subset selection (Hall, 1998). RandomForest uses bagging to create an ensemble of semi-random decision trees (Breiman, 2001), and OrdinalClassClassifier is a simple meta learner that treats the class as an ordinal quantity and creates several binary classification problems based on this assumption (Frank and Hall, 2001). J48 is used to tackle these binary classification problems.

Decorate (Melville and Mooney, 2003) builds an ensemble of decision trees by creating artificial data to obtain an ensemble of diverse base classifiers. Again, the J48 decision tree learner is used as the base learner. MultiLayerPerceptron implements the classic feed-forward neural network (Rumelart et al., 1986; Bishop, 1995), using one hidden

layer and as many hidden units as there are output units. NBTree builds a decision tree with naive Bayes models at the leaves (Kohavi, 1996), NNGE implements non-nested generalized exemplars, a variant of Salzberg's algorithm for building nearest-neighbour classifiers with rectangular generalizations (Salzberg, 1991), and RandomCommittee builds an ensemble of ten semi-random decision trees (without using bagging). It is not surprising that an ensemble of ten trees ranks before a single random tree, as implemented by RandomTree (second scheme from the bottom in Table 2).

FilteredClassifier, in default mode, as used here, and J48 differ only in that the former uses the supervised MDL-based discretization scheme mentioned above on all numeric attributes prior to building the decision tree. The results show that discretization makes very little difference. MultiClassClassifier and Logistic are also very similar, because, in default mode, the former uses logistic regression as the base learner and applies the one-vs-all technique for tackling this multi-class problem (Rifkin and Klautau, 2004). Ridor is a ripple-down rule learner (Witten and Frank, 2005). Part builds a decision list using partial decision trees (Frank and Witten, 1998), and JRIP implements the RIPPER rule learning algorithm (Cohen, 1995). It is interesting to see that all these more sophisticated rule learning algorithms ranked lower on this data, while the simple schemes that consider a single attribute, like DecisionStump or OneR, ranked higher.

KStar is a nearest-neighbor scheme that uses an entropy-based distance function (Cleary and Trigg, 1995) . IBk is the simple nearest-neighbor classifier based on Euclidean distance (Witten and Frank, 2005). The latter binarizes all nominal attributes in the data. The low rankings of these schemes, and the high ranking of naive Bayes and single-attribute-based classifiers like OneR, are another indication that there were many irrelevant attributes in this data that can confound distance-based learning algorithms.

The last algorithm in Tables 1 and 2 is ZeroR, which simply predicts the majority class in the training data. Hence its value for the Kappa statistic is zero, equating to no improvement compared to a random predictor. It is included in the results solely to serve as a base-line, and, given that it is ranked at the bottom, it appears that all other learning algorithms that were investigated managed to extract some information from the data to a more or less successful degree.

The length of the list in Table 3 for the regression experiment, where the target value was numeric, is shorter than in Tables 1 and 2. This is due to the fact that Weka has fewer implementations of regression algorithms than classification algorithms. The expert system is ranked fourteen out of twenty. Number one in the ranking is LeastMedSq, which attempts to find a linear regression function by minimizing the median squared error on the training data (Witten and Frank, 2005). It is usually more robust to outliers than LinearRegression, which is the sixth in ranking. The latter performs least-squares linear regression, with some built-in attribute selection heuristics (Witten and Frank, 2005).

M5Rules and M5P are very similar in ranking. M5P builds model trees, mentioned above in the classification case. M5Rules builds a decision list by repeatedly calling M5P in a separate-and-conquer fashion, and converting one leaf into an if-then-rule in each iteration (Holmes et al., 1999). SMOReg builds a linear support vector machine for regression (Smola and Schoelkopf, 1998) and performs similar to the other schemes that build linear models, LeastMedSq and LinearRegression.

Bagging has already been described above in the classification case. As in the classification case, REPTree is used as the base learner. In the regression case, REPTree generates a regression tree instead of a decision tree, and thus bagging can be applied to regression problems using this configuration. DecisionTable has also been mentioned above. As opposed to the classification case, where the best-first search for a good subset of attributes aims to minimize the classification error, the squared error is minimized instead. LWL, ConjunctiveRule, and DecisionStump have also already been discussed in the classification case. The only difference, when applied to regression, is that the squared error on the training data is minimized instead of an entropy-based criterion. For example, DecisionStump builds a one-level regression tree based on this criterion. As in the classification case, it is interesting to see that very simple models perform relatively well on this data: linear models that work with sets of attributes, as well as even simpler models like decision stumps that are based on a single attribute.

RegressionByDiscretization implements a simple technique that facilitates the application of classification algorithms to regression problems. To this end, it simply discretizes the numeric target using equal-width discretization. By default, ten intervals are used for the discretization and the J48 decision tree learner is applied as the classification schemes. Predictions are made by combining the observed mean values for each interval using the class probability estimates obtained from the classification algorithm. It is interesting to note that this method performs identically to REPTree, which builds a tree structure that models the target directly.

Additive regression can be viewed as a boosting-like algorithm for regression. It builds an ensemble of predictors by greedily minimizing the squared error of the ensemble (Friedman et al., 2000). In each iteration it applies a base model that is fitted to the residuals that remain from the previous iteration. By default, DecisionStump is used as the base learner to build one-level regression trees as the base models in each iteration, and ten iterations of the algorithm are performed. The results show that AdditiveRegression does not actually manage to improve much on simply building a singe regression stump.

Considering the group of regression schemes that ranked lower than the expert system, we can identify several techniques that build models representing non-linear regression functions. RBFNetwork builds a radial basis function network, mentioned above in the classification case, for regression, and KStar and IBk have also already been mentioned in the classification case. ZeroR simply predicts the mean target value observed in the training data. Surprisingly both the IBk nearest-neighbor classifier and the multi-layer perceptron perform worse than this simple technique, indicating that they overfitted the training data.

Again, it is important to remember that only the first seven algorithms in Table 3 had a statistically significant advantage over the expert system according to one-sided hypothesis testing with ($\alpha = 0.05$). This is unlike the results of the classification experiment where not a single machine learning model showed any statistically significant advantage over the expert system at the same level of confidence, both according to the hit ratio and the Kappa statistic.

6. Conclusions

Similar to previous comparisons between machine learning models and expert systems' accuracy, this research was based upon a single expert system. Therefore, one must be very careful while interpreting its results.

Unlike previous reports, this experiment has shown, though, a case where a statistically significant, accuracy-wise, advantage of machine learning models over an expert system could only be established in the regression case. In our study, we attempted to simulate a naïve user of machine learning, who is not an expert in machine learning algorithms and does not know which parameters to tune to improve results. We believe that this is a more realistic scenario than those considered in other papers (and it also counteracts overfitting to the dataset at hand). In regression, seven machine learning models had a statistically significant advantage over the expert system. In contrast to our initial research hypothesis, and unlike any previous publication - no statistically significant advantage of any machine learning model could be established in the classification case. These results may seem contradictory at a first glance. Should the regression or the classification experiment results be adopted?

In our view, machine learning models have failed to show a convincing advantage over the expert system in this particular case. The reason we think so, despite of the seven out of nineteen clear-cut "wins" of machine learning models in the regression experiment, is due to the nature of this particular domain. Although represented by numbers, the real nature of the output was ordinal. This is due to the fact that, for example, even the best expert cannot distinguish between a credit worthiness of, say, 0.83 and 0.82. Experienced credit officers can, however, compare clients with each other, saying that one is "better", "worse" or "similar" to another. In other words, representing ordinal value with numbers does not reflect the true nature of the class. It is ordinal. Not numeric. For this reason, we tend to have a greater faith in the classification results, where no statistically significant advantage of machine learning model over the expert system could be established.

There is a bright side, however, to these somehow "disappointing" results from the point of view of machine learning models. A lot of effort and resources were invested in developing this particular expert system. It was built over a period of several months and a couple of expensive man-years were invested in it. Nevertheless, in only a matter of a few days' work, we have managed to generate dozens of machine learning models that showed no statistically significant inferiority, or disadvantage, relative to the expert system. This is quite impressive, considering the fact that no real attempt has been made to optimize the machine learning models by finding optimal values for their parameters. On the other hand, if we had done so, one could have rightfully argued that the expert system's performance could have been improved too. As time passes and more feedback is accumulated, one usually modifies the rule-base as well.

It was also quite surprising to see that the use of the Kappa statistic instead of the hit ratio as an accuracy meter promoted the ranking of the expert system from the twenty first position to the third. The Kappa statistic compensates for classifications that may be due to chance, while the hit ratio does not. For this reason, we consider the Kappa statistic a more reliable meter for classification problems than the hit ratio. As this case study demonstrates, the meter that is being used to measure accuracy can and does make a difference. We therefore encourage all classification results to include the Kappa statistic.

The expert system that has been studied here has not shown any statistically significant advantage over machine learning models. So the question arises, why it was built in the first place, when there were cheaper and faster alternatives in machine learning? There were two major reasons: A. No suitable data was available for using any machine learning model when the expert system was built. The data was only accumulated during the expert system construction phase, for testing its performance. B. No less important is the fact that the experts felt more comfortable with the rules they defined themselves, and they were less trusting in machine learning models, upon which, they felt, they had little or no control. According to the findings of our experiment - one cannot claim that they were either right or wrong as far as accuracy is concerned.

Clearly, more expert systems in various domains need to be tested against machine learning models before one can reliably conclude which of the approaches gives more accurate results, and under which conditions. This is not an easy task, though. Hopefully, more direct comparisons between "mind crafted" expert systems and machine learning models will be published in the future.

References

C.G. Atkenson, A.W. Moore, and S. Schaal (1997) "Locally weighted learning," Artificial Intelligence Review, 11, pp. 11-73.

C.M. Bishop (1995) "Neural networks for pattern recognition", Oxford, University Press.

L. Breiman (1996) "Bagging Predictors", Machine Learning, 24, pp. 123-140.

L. Breiman (2001) "Random Forests", Machine Learning, 45, pp. 5-32.

D. V. Cicchetti and A.R. Feinstein, 1990, "High agreement but low kappa: Resolving the paradoxes", Journal of Clinical Epidemiology 43, 6, pp. 543-558, . In two parts.

J.G. Cleary, L.E. Trigg (1995) "K*: An Instance-based Learner Using an Entropic Distance Measure", Proc 12th International Conference on Machine Learning, pp. 108-114, Morgan Kaufmann.

R. J. Cook (1998) "Kappa and Its Dependence on Marginal Rates". In: Encyclopedia of BioStatistics, P. Armitage, T.Colton, (Eds), pp. 2166-2168. Wiely, NY.

J. A., Cohen (1960) "Coefficient of Agreement for Nominal Scales", Educational and Psychological Measurement, pp. 37-46.

W. W. Cohen (1995) "Fast Effective Rule Induction", Proc Twelfth International Conference on Machine Learning, pp. 115-123, Morgan Kaufmann.

R. H. Creecy, B.M. Masand, S.J. Smith and D.L. Waltz (1992) "Trading MIPS and Memory for Knowledge Engineering", Communications of the ACM, 35, 8, pp. 48- 83.

O. Daren, G. Warner, A. White, and C. Bessant (2005) "Comparison of machine learning techniques and expert systems for the determination of microarray quality", Poster presentation no. G-A6, Computational Biology, Madrid. www.ecc05.org.

P. Domingos and M. Pazzani (1997) "On the optimality of the simple Bayesian classifier under zero-one loss", Machine Learning, 29, pp. 103-130.

U.M. Fayyad and K.B. Irani (1993) "Multi-interval discretization of continuous- valued attributes for classification learning , Proc Thirteenth International Joint Conference on AI, pp. 1022-1027, Morgan Kaufmann.

E. Frank and I.H. Witten (1998) "Generating Accurate Rule Sets Without Global Optimization", Proc Fifteenth International Conference on Machine Learning, pp. 144-151, Morgan Kaufmann.

E. Frank and Mark Hall (2001) "A Simple Approach to Ordinal Classification", Proc Twelfth European Conference on Machine Learning, pp. 145-156, Springer Verlag.

Y. Freund and R.E. Schapire (1996) "Experiments with a new boosting algorithm", Proc Thirteenth International Conference on Machine Learning, pp. 148-156, Morgan Kaufmann.

J. Friedman, T. Hastie and R. Tibshirani (2000) "Additive logistic regression: a statistical view of boosting", Annals of Statistic, 28, pp. 337-407.

M.A. Hall (1998) "Correlation-based Feature Subset Selection for Machine Learning", PhD thesis, Department of Computer Science, University of Waikato, Hamilton, New Zealand.

G. Holmes, M. Hall and E. Frank (1999) "Generating Rule Sets from Model Trees", Proc Twelfth Australian Joint Conference on Artificial Intelligence, pp. 1-12, Springer Verlag.

R. Holte (1993) "Very simple classification rules perform well on most commonly used datasets", Machine Learning, 11, pp. 63-91.

R. Kohavi (2005) "The Power of Decision Tables", Proc Eighth European Conference on Machine Learning, pp. 174-189, Springer Verlag.

R. Kohavi (1996) "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid", Proc Second International Conference on Knowledge Discovery and Data Mining, pp. 202-207, ACM Press.

N. Landwehr, M. Hall and E. Frank (2005) "Logistic model trees", Machine Learning, 59, pp. 161-205.

T.S. Lim, W.Y. Loh, and Y.S. Shih (2000) "A Comparison of Prediction Accuracy, Complexity and Training Time of Thirty-three Old and New Classification Algorithms", *Machine Learning*, 40, pp. 203-229.

P. Melville, R.J. Mooney (2003) "Constructing Diverse Classifier Ensembles Using Artificial Training Examples", Proc Eighteenth International Joint Conference on Art ificial Intelligence, pp. 505-510, Morgan Kaufmann.

R. S. Michalski and R.L. Chilausky (1980) "Knowledge acquisition by encoding expert rules versus computer induction from examples: A case study of involving soybean pathology" International Journal of Man-Machine Studies, 12, pp. 63-87. Republished in Int. J. of Human-Computer Studies (same journal) , 51,2, August 1999, pp. 239-263.

M. Musen, (1989) "Automated support for building and extending expert models", Machine Learning, 4, pp. 347-375.

J. Platt (1998) "Fast Training of Support Vector Machines using Sequential Minimal Optimization." In Advances in Kernel Methods- Support Vector Learning, eds. Schoelkopf and C. Burges and A. Smola, pp. 185-208, MIT Press.

J.R. Quinlan (1987) "Simplifying decision trees", In Knowledge Acquisition for Knowledge-Based Systems, eds. B. Gaines and J. Boose, pp. 239-252, Academic Press.

J.R. Quinlan (1992) "Learning with Continuous Classes", Proc. Fifth Australian Joint Conference on Artificial Intelligence, pp. 343-348, World Scientific.

J.R. Quinlan (1993) "Programs for machine learning", Morgan Kaufmann.

R. Rifkin and A. Klautau (2004) "In Defense of One-Vs-All Classification" Journal of Machine Learning Research, 5, pp. 101-141.

D.E. Rumelhart and J. L. McLelland, (Eds.) (1986) "Parallel Distributed Processing Learning: Internal Representation by Error Propagation" Vol.1, MIT Press, Cambridge, Mass.

S. Salzberg (1991) "A nearest hyperrectangle learning method," Machine Learning, 6, pp. 251-276.

A.J. Smola and B. Schoelkopf (1998) "A Tutorial on Support Vector Regression", In NeuroCOLT2 Technical Report Series, NC2-TR-1998-030.

G.I. Webb (2000) "MultiBoosting: A Technique for Combining Boosting and Wagging", Machine Learning, 40, pp. 159-196.

I.H Witten and E. Frank (2005) "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann.