Universal Prediction Model for Construction Litigation

Thaveeporn Pulket[1] and David Arditi[2]

Abstract: Construction litigation expenditures have increased considerably over the years. An universal prediction model (UPM) was developed to predict the outcome of construction litigation and, hence, encourage settlements out of court. The study was conducted by using 151 Illinois circuit court cases filed in the period 1987–2005. UPM consists of data consolidation, attribute selection, hybrid classification, and performance assessment. A code was written to automate the entire process in the Waikato environment for knowledge analysis. UPM is versatile and scalable. The findings resulted in a higher prediction rate than those obtained in previous studies. The system proved to be quite robust and fast. If the outcome of construction litigation can be predicted with reasonable accuracy and reliability, all parties involved in the construction process could save considerable money and time.

DOI: 10.1061/(ASCE)0887-3801(2009)23:3(178)

CE Database subject headings: Litigation; Court decision; Artificial intelligence; Predictions; Hybrid methods; Construction industry.

Introduction

Construction litigation has become commonplace in numerous construction projects, particularly in large contracts. Disputes are caused by a variety of reasons including design changes, changed site conditions, delays, and acceleration orders, to name but a few. Construction litigation expenditures have increased at an average rate of 10% (Pena-Mora et al. 2003) per year. Several alternative dispute resolution methods such as arbitration, mediation, and dispute review boards have been developed as an alternative to lengthy and costly litigation. Many attempts have been made by researchers to reduce the number of claims and disputes and mediate between the parties, hence, avoiding litigation (Kassab et al. 2006; Ren et al. 2003; AbouRizk and Dozzi 1993; Kilian and Gibson 2005; Diekmann and Girard 1995; Yiu et al. 2006; Russell 1996; Elazouni 2006). Artificial intelligence-based models have been used extensively to solve problems in various civil engineering domains (e.g., transportation, hydraulics, and soil mechanics). Some of these models have made use of hybrid systems and obtained improved results [e.g., neural-expert system that predicts the strength of concrete developed by Gupta et al. (2006)]. There have been only three attempts to reduce the frequency and severity of construction litigation by developing artificial intelligence-based models to predict the outcome of court decisions, one using artificial neural networks (Arditi et al. 1998), another using case-based reasoning (Arditi and Tokdemir 1999), and the final one using boosted decision trees (Arditi and Pulket 2005) on construction-related Illinois court cases, reaching prediction rates of 66.67, 83.33, and 87%. The first two of these models used a single classifier to make the predictions, while the third one made use of boosting, an enhancing algorithm. The idea of an integrated system composed of a multitude of combinations of several preprocessing methods and several hybrid classification systems is explored in this study in order to optimize the prediction of the outcome of court cases in the construction litigation domain.

A universal prediction method (UPM) was developed to be a comprehensive methodology to solve prediction problems by exploring the combination of as many preprocessing methods and as many hybrid classifiers as possible. It is expected that prediction rates can be improved by developing new methodologies such as UPM.

Universal Prediction Model

Experience and past literature reviews show that no single machine learning scheme is appropriate to all data mining problems (Witten and Frank 2005). Real data sets vary, and to obtain accurate models, the bias of the learning algorithm must match the structure of the domain. Data mining is an experimental science (Witten and Frank 2005). The concept of a universal learner proposed in this study is an ideal approach that would be applicable to all problems in all domains.

The UPM developed in this study is composed of three main processes, namely, preprocessing, classification, and assessment (Fig. 1). The model is constructed using the Waikato environment for knowledge analysis (WEKA) (Witten and Frank 2005) and implemented in the construction litigation domain. WEKA is a collection of machine learning algorithms and data preprocessing tools and provides an environment for experimental data mining. WEKA is available from the Computer Science Department of the University of Waikato, New Zealand.

Preprocessing

Preprocessing consists of two functions as shown in Fig. 1. First, the dataset is appropriately consolidated by using several algo-

[1]Formerly, Graduate Student, Dept. of Civil, Architectural, and Environmental Engineering, Illinois Institute of Technology, Chicago, IL 60616. E-mail: pulktha@iit.edu

[2]Professor, Dept. of Civil, Architectural, and Environmental Engineering, Illinois Institute of Technology, Chicago, IL 60616. E-mail: arditi@iit.edu

Note. Discussion open until October 1, 2009. Separate discussions must be submitted for individual papers. The manuscript for this paper was submitted for review and possible publication on March 12, 2008; approved on October 15, 2008. This paper is part of the ***Journal of Computing in Civil Engineering***, Vol. 23, No. 3, May 1, 2009. ©ASCE, ISSN 0887-3801/2009/3-178–187/$25.00.

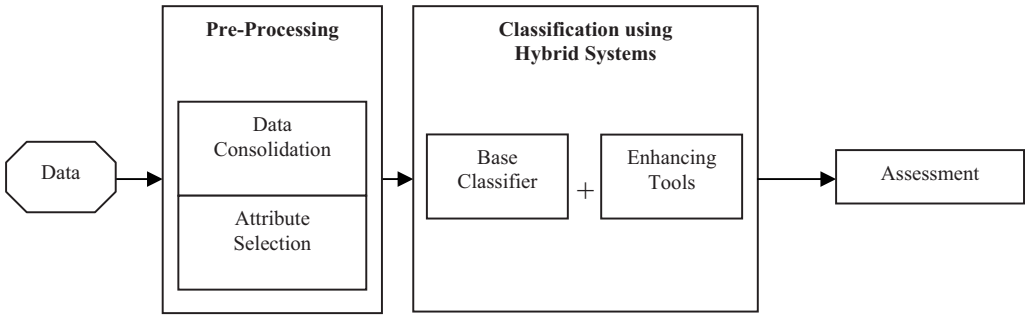

Fig. 1. Framework of the universal prediction model

rithms and methods. This step is performed to ensure the appropriate data format required for a particular learning scheme. Next, the attribute selection process searches the space of attribute subsets and selects the most relevant attributes in the dataset.

Data Consolidation

Data consolidation attempts to make the data more efficiently manageable by machine learning tools because real data are often low in quality. Indeed, data may come from different sources at different times. The data may have to be extracted from text, assembled, integrated, and organized. The data have sometimes not been gathered expressly for the purpose in mind. When originally collected, some of the fields probably did not matter and were not filled properly. Provided that it does not affect the original purpose of the data, there is no incentive to correct them. However, when the same data are used for a specific purpose, the errors and omissions may suddenly start to assume great significance. Most data sets encountered in practice also contain missing values. The significance of missing values has to be carefully considered. Missing values may occur for several reasons, such as malfunctioning measurement equipment, changes in design during data collection, and collation of several similar but not identical data sets.

Data consolidation involves the application of a filter algorithm to the data sets at hand. If the filter is supervised, i.e., if it uses class values to derive intervals for discretization, applying it to the test data may bias the results. To avoid the occurrence of bias, the unsupervised attribute filters are used in this research.

Attribute Selection

Each individual, independent case that provides input to machine learning is characterized by its values on a fixed, predefined set of features called attributes. Most machine learning algorithms are designed to learn to use the most appropriate attributes for making their decisions. Experiments with a decision tree learner have shown that adding to standard data sets a random binary attribute generated by tossing an unbiased coin affects classification performance, causing it to deteriorate by about 5–10% (Witten and Frank 2005). This happens because at some points in the trees that are learned, the irrelevant attribute is invariably chosen to branch on, causing random errors when test data are processed.

Because of the negative effect of irrelevant attributes on most machine learning schemes, it is possible to precede learning with an attribute selection stage that strives to eliminate all but the most relevant attributes. The best way to select relevant attributes is manually, based on a deep understanding of the learning problem and what the attributes actually mean. However, automatic methods can also be useful. Deleting unsuitable attributes improves the performance of learning algorithms, and speeds them up, although this may be outweighed by the computation involved in attribute selection. More importantly, dimensionality reduction yields a more compact, more easily interpretable representation of the target concept, focusing the user's attention on the most relevant variables.

Machine learning algorithms can be used for attribute selection. For instance, the decision tree algorithm can be applied to the full dataset, and then only those attributes that are actually used in the tree are selected. When a decision tree is used for attribute selection, it will not have any effect if the base classifier is also a decision tree, but it may have a positive effect if the base classifier is a different learning algorithm. Some algorithms are notoriously susceptible to irrelevant attributes, and their performance can be improved by using a decision tree builder as a filter for attribute selection first.

Classification Using Hybrid Systems

"Hybrid" means something made up of a mixture of different elements. In this study, a hybrid intelligence system consists of at least one base classifier and other additional components (see Fig. 1). Base classifiers include decision trees (based on the divide and conquer approach), rule learning schemes (that take each class in turn and seek a way of covering all instances in it, at the same time excluding instances not in the class), and function learning schemes (mathematical equations written in a reasonably natural way).

The addition to a base classifier can be in the form of meta-learner applications or other classifiers (ensemble methods). Adding enhancing tools such as meta-learner applications or using ensemble methods is likely to improve the performance of a base classifier, and, hence, enhance the predictive performance of the system compared to using a single model.

Meta-Learner Approach

Meta learners include tools such as boosting and bagging that enhance the performance of a base classifier. Bagging is a variance reduction scheme, while boosting primarily reduces the bias of the base procedure. There are two major differences between bagging and boosting. First, boosting changes adaptively the distribution of the training set based on the performance of previously created classifiers, while bagging changes the distribution of the training set stochastically. Second, boosting uses a function of the performance of a classifier as a weight for voting, while bagging uses equal weight voting. Boosting algorithms are considered stronger than bagging on noise-free data; however, bag-

ging is much more robust than boosting in noisy settings. While boosting reduces both bias and variance, bagging mainly reduces variance (Webb 2000). MultiBoost AB is an algorithm that combines boosting with bagging. It retains boosting's bias reduction ability while adding bagging's effective variance reduction. However, whereas bagging uses resampling to get the data sets for training and producing a weak hypothesis, MultiBoost AB uses wagging, i.e., reweighting for each training example, pursuing the effect of bagging in a different way (Kotsiantis and Pintelas 2004).

Ensemble Approach

The ensemble approach includes tools such as stacking, grading, and voting that manipulate the output of multiple models. Unlike bagging and boosting that are applied directly to a base classifier, the ensemble approach involves running multiple models, and applying tools such as stacking, grading, and voting to extract enhanced predictions. The procedure involves estimating the expected error of combinations and choosing the combination with the lowest error and, hence, the highest prediction capability.

The basic idea of *Stacking* is to use the predictions of the base classifiers as attributes in a new training set that keeps the original class labels (Seewald 2002). Stacking learns from the outputs of the base classifiers (e.g., decision trees, JRip) by using another learning algorithm (Wolpert 1992) that makes use of class probability distributions. A variation of stacking called StackingC makes use of multi response linear regression rather than class probability distributions and has proved to be more efficient than basic stacking (Seewald 2002). StackingC was used in this research.

Grading is a technique developed by Seewald and Fürnkranz (2001) who have made use of the principles originally set by Bay and Pazzani (2000). Grading trains a classifier for each of the original base classifiers on a training set that consists of the original cases but with class labels that encode whether the prediction of this classifier was correct on this particular case. So the original dataset is transformed into a two-class dataset with new values that encode whether the base classifier was able to correctly predict this case in an internal cross-validation or not (Seewald and Fürnkranz 2001). Hence, in contrast to stacking, grading leaves the original cases unchanged, but instead modifies the class labels.

In *voting*, instead of giving its entire vote to the class it considers to be most likely, each base classifier is allowed to split its vote according to its estimate of the class probability distribution for the particular case. Voting adds up the distributions of all base classifiers, renormalizes the sum, and chooses the class with highest probability afterwards (Kuncheva 2004).

Validation

Classifier accuracy is important in that it allows one to evaluate how accurately a given classifier will label future data, that is, data on which the classifier has not been trained. Using training data to derive a classifier and then using the same data to test for accuracy can result in misleading overoptimistic estimates due to overspecification of the learning model to the data. The hold out and cross-validation methods are two common techniques for assessing classifier accuracy. In the hold out method, the data are randomly partitioned into two independent sets, training and test sets. Typically, two-thirds of the data are allocated to the training set, and the remaining one-third to the test set (Nashvili 2006). The training set is used to derive the classifier, whose accuracy is assessed with the test set. In k-folds cross-validation, the initial data are randomly partitioned into k mutually exclusive subsets or "folds" $(S_1, S_2, \ldots, S_k)$ each of approximately equal size. Training and testing are performed k times. In iteration, i the subset S_i is reserved as the test set, and the remaining subsets are collectively used to train the classifier. Accuracy is the overall number of correct classifications from the k iterations, divided by the total number of samples in the initial data. In stratified cross-validation, the folds are stratified so that the class distribution of the samples in each fold is approximately the same as that in the initial data. A stratified tenfold cross-validation is recommended for assessing classifier accuracy due to its relatively low bias and variance (Han and Kamber 2001).

The results are analyzed and compared to pick the models that perform better with the construction litigation dataset.

Assessment

In this process, prediction systems are compared and evaluated on the basis of four performance measures: prediction accuracy, robustness, speed, and interpretability. If the top priority is prediction accuracy, the hybrid combination that gives the highest rate of prediction is selected. If the top priority is robustness, the hybrid combination that has the most consistent results with crisp and noisy data is selected. This process is extended to speed and interpretability too. If on the other hand, a combination of priorities is to be preferred, then the assessment is made accordingly.

- Prediction accuracy: this criterion is assessed based on how correctly a model classifies instances measured by the mean absolute error. This metric shows the predictive performance of different models by indicating the level of accuracy in terms of percent of test cases predicted correctly. The k-folds cross-validation method is implemented in the experiments.
- Robustness: noisy and missing values in the dataset can affect the robustness of different learning algorithms in different ways. Comparing the outcomes of different models in the face of noise and missing values will give an indication of the robustness of these models. This criterion is assessed based on the model's sustainability from external distraction factors such as noise in data and missing values. Preprocessing filters are used to manipulate the dataset creating noisy information and unknown attributes. The model that can sustain, or in other words, minimize its prediction error under these circumstances is considered to be more robust.
- Speed: this criterion is assessed based on the computation time of the model. This metric reflects the complexity of the model that is being used. For example, it is obvious that a neural network model will take longer to run than a decision tree model due to the demanding computational processes of its multilayer structure compared to the straight forward branches in a decision tree model.
- Interpretability: this criterion is assessed based on the output of the models. This factor is subjective and depends heavily on the structure of different algorithms in different models. For example, decision tree-based algorithms produce a tree, weights, and branch probabilities; rule-based algorithms produce rules that lead to the class; yet neural network-based algorithms return incomprehensible results that cannot be interpreted in the real world meaning. This metric is useful when trying to put together practical rules and guidelines relative to the problem domain.

Table 1. Lists of Methods Used in UPM

Preprocessing		Classification using hybrid systems	
Data consolidation	Attribute selection	Base classifiers	Enhancing tools
Unsupervised attribute filters	Subset evaluation methods	Decision trees learning schemes	Meta learners
Add noise	Correlation based	J4.8 (C4.5)	Boosting
Normalize	Classifier subset	REP Tree	Bagging
Remove useless	Consistency subset		MultiBoost AB
Replace missing values	Search methods	Rule learning schemes	Ensemble methods
Class order	Best first	JRip	Grading
	Genetic search	Part	StackingC
	Greedy stepwise		Vote
	Rank search		

Methodology

Scope of the Model

Although WEKA is a resourceful environment, only a limited number of tools were selected for use in the model. Bound by the amount and characteristics of the dataset at hand, some tools are specifically applicable and some are more suitable than others. The criteria of tool selection to form the model are described in this section:

- Data consolidation—all attributes in the dataset are expressed by nominal values. Since the bias in the model is to be minimized, only unsupervised filters are applicable. Since nominal values cannot be changed into other types, the transforming filters were discarded. Add noise was selected to test the robustness of the model. As seen in Table 1, the following self-explanatory filters were also selected: normalize, replace missing values, reorder, and remove useless.
- Attribute selection—subset evaluation methods are more accurate but more time consuming than single attribute evaluation methods. Therefore, the following attribute selection methods were used in the study (see Table 1): correlation-based feature subset evaluator, classifier subset evaluator, and consistency subset evaluator. On the other hand, only some of the search methods are compatible with these subset evaluation methods and, therefore, these search methods were singled out for use in this study (see Table 1): best first, genetic search, greedy stepwise search, and rank search.
- Base classifiers and enhancing tools—the criteria used for selecting base classifiers out of all the tools available in the WEKA environment include potentially fast speed with promising results. Past research indicates that some tools have been proven outdated and have been replaced by newer, improved algorithms. Sometimes, more than one variation of the same algorithm have been developed, tested, and compared with the original. In such cases, one of the variations has proved to be superior to all others, hence, replacing the older variations. Using these principles, the following base classifiers were selected for use in the study (see Table 1): J4.8, REPTree, JRip, and PART. J4.8 is the name of a Java class that generates an unpruned or pruned C4.5 decision tree, where C4.5 is an original well known algorithm for a decision tree learning scheme (Quinlan 1993). REPTree builds a decision/regression tree using information gain/variance reduction and prunes it using reduced-error pruning with backfitting. REPTree can be considered to be a faster variant of the decision tree learning scheme used in C4.5. JRip is a Java-based version of the RIPPER algorithm (repeated incremental pruning to produce error reduction) developed by Cohen (1995). PART, partial decision tree, is a method that combines C4.5 and RIPPER to avoid global optimization and to produce accurate and compact rule sets (Frank and Witten 1998).

Boosting, bagging, MultiBoostAB, grading, StackingC, and vote were used as enhancing tools. These tools were briefly discussed in the preceding section. Function learning schemes were excluded because of their excessively long computational time.

Structure of UPM

The framework of UPM is illustrated in Fig. 1. First, six data sets including the original file were generated using the data consolidation methods listed in Table 1. Twelve attribute selection methods were set by using the three subset evaluation methods in combination with the four search methods listed in Table 1. A total of 72 data sets were, hence, created using these 12 attribute selection methods on the six data sets. These 72 data sets were subjected to the four base classifiers and 12 hybrid classifiers generated by using the four base classifiers in combination with the three meta learners listed in Table 1, creating a total of 1,152 experiments.

The combinations of the four base classifiers used in the ensemble methods can be expressed by a combinatorial equation as

$$C_k^n = \frac{n!}{k!(n-k)!} \tag{1}$$

where C=Numbers of combinations, n=number of objects, and k=number to be chosen.

In this case, the combinations are $C_4^4+C_3^4+C_2^4$ since C_1^4 is the use of only one classifier and is not considered. Hence, the final combinations are 11 as shown in the following equation:

$$\begin{aligned} C_4^4 + C_3^4 + C_2^4 &= \frac{4!}{4!(4-4)!} + \frac{4!}{3!(4-3)!} + \frac{4!}{2!(4-2)!} \\ &= 1 + 4 + 6 \\ &= 11 \end{aligned} \tag{2}$$

The same 72 data sets were also subjected to these 11 combinations of the four base classifiers used in combination with the three ensemble methods listed in Table 1, creating a total of 2,376 experiments.

Judging from a pilot study where only 16 experiments were conducted manually, a fair amount of time and tedious processes

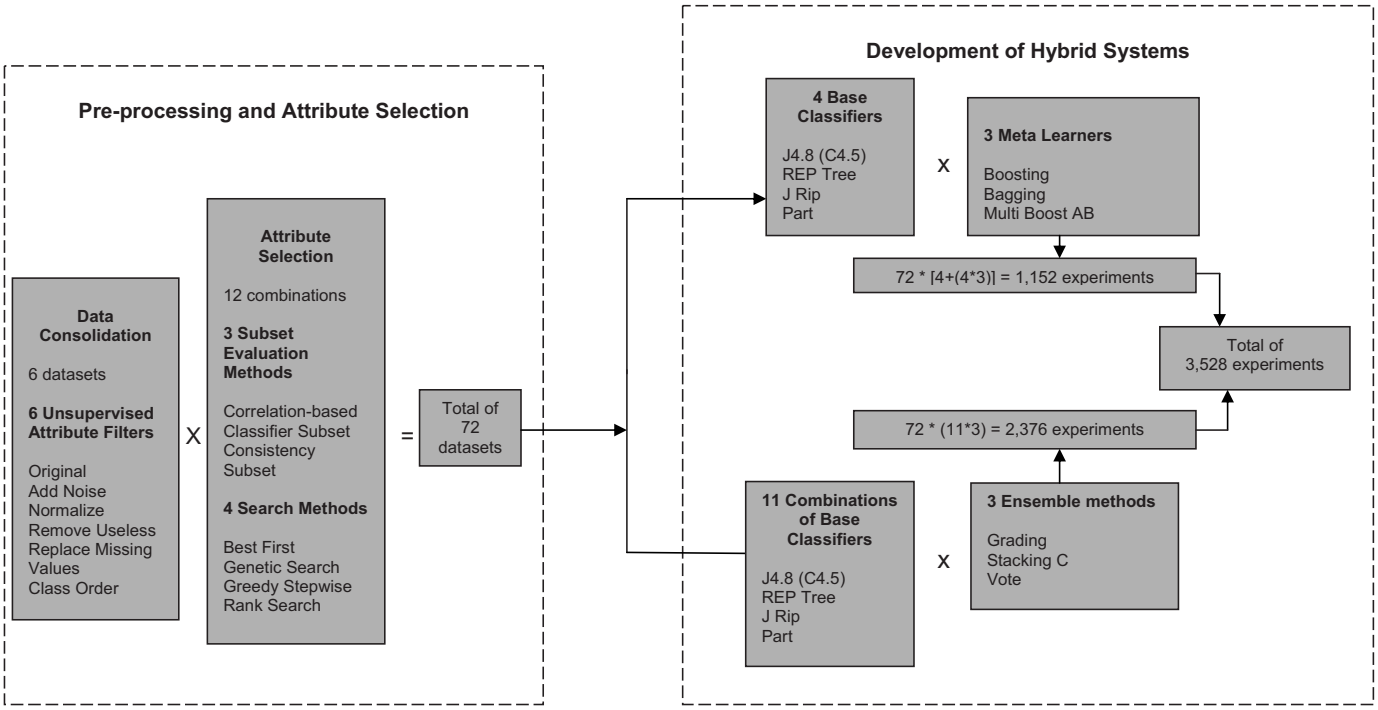

Fig. 2. UPM framework

were required to organize the data sets, create the hybrid classifiers, and run the experiments. Automating this process was found to be critical in order to generate the results effectively, accurately, and in a timely manner.

As seen in Fig. 2, a total of 3,528 experiments were performed in this study. An automated system was, therefore, developed such that a user needs only to enter the raw data and the processes follow each other automatically. The automation tool was developed in Microsoft Access as a script generator systematically mimicking all the manual processes needed to be done. Tables recording all the findings were created.

Three major tables were generated for storing the cases, containing the respective execution codes written for the data consolidation process, the attribute selection process, and the hybrid classifier manipulations. In these tables, a row was used to store different tools or combination of tools. To enhance scalability, a column called "Check" was placed in the tables to enable/disable the tools or combination of tools. A tool is used only when a check mark is present in the "check" column in that tool's row. This scalable design benefits experimenters not only for comparison but educational purposes too. Users can choose to consider any and all tools and conduct a small set of tests with only a few tools, or conduct a large set of experiments that involves many tools. The back end of this scalable design relies on query tables (Q tables), which basically generate the execution codes according to whether there is a check mark in the "check" column.

The row in every table is customizable. Users can modify the information in a row by adding or deleting any tools they wish to use. However, one has to obey the same format rules so that the generator can work without error.

A filing system was developed to systematically identify the combinations used in UPM. Cases were filed using the following system:

AA-XXYYZZ.arff

where: AA=output file of the process: 1a for data consolidation, 1b for attribute selection, or 2ab for hybrid classification, XX =ID of data consolidation method, YY=ID of attribute selection method, and ZZ=ID of hybrid classifier.

For example, the following file, 1A-010232.arff, is the output file of the data consolidation process (1a) and is set to use the raw data (ID=01), the CFS+BEST attribute selection method (ID =02), and to use grading with combination 1 (ID=32). The generator keeps a record of the outputs in all three procedures, i.e., data consolidation, attribute selection, and hybrid classification.

Statistical data are generated to provide a performance report for each file by means of a script written in Microsoft Excel

B	C	D	E	F	G	H	I	J	K	L
File Name	Correctly	Incorrectly	Kappa	Mean	Root Mean	Relative	Root Relative	Total	Correct Cases	Incorrect Cases
2AB-010201.arff	90.0662	9.9338	0.8542	0.0453	0.1686	19.4811	49.6312	151	136	15
2AB-010202.arff	81.457	18.543	0.7309	0.0794	0.2233	34.1169	65.7312	151	123	28
2AB-010203.arff	90.7285	9.2715	0.865	0.0464	0.1647	19.9562	48.4638	151	137	14
2AB-010204.arff	91.3907	8.6093	0.8734	0.0438	0.1641	18.8287	48.3001	151	138	13
2AB-010207.arff	94.0397	5.9603	0.9131	0.0325	0.1388	13.971	40.8661	151	142	9
2AB-010208.arff	90.0662	9.9338	0.8558	0.0703	0.1743	30.2112	51.2969	151	136	15
2AB-010209.arff	94.702	5.298	0.9228	0.0384	0.1375	16.4954	40.4815	151	143	8

Fig. 3. Sample of final results

Table 2. Top 10 Prediction Rates of Original Dataset

Filenames	Prediction rate	Kappa statistic	RMSE	Total cases	Correct cases
2AB-011009.arff	90.9091	0.8689	0.1846	132	120
2AB-011107.arff	90.9091	0.87	0.1732	132	120
2AB-010860.arff	90.1515	0.8582	0.185	132	119
2AB-010862.arff	90.1515	0.8584	0.1945	132	119
2AB-011110.arff	90.1515	0.8583	0.1775	132	119
2AB-010654.arff	89.3939	0.8468	0.1879	132	118
2AB-010655.arff	89.3939	0.8468	0.1888	132	118
2AB-011161.arff	89.3939	0.8483	0.1961	132	118
2AB-010604.arff	88.6364	0.8356	0.1912	132	117
2AB-010640.arff	88.6364	0.8356	0.1946	132	117

macro. This script was developed by making use of the Helmer (2005) directory file listing utility. A sample of the statistical information in the final table is presented in Fig. 3.

Findings and Discussion

The study used two data sets to conduct the experiment, the original dataset of 132 cases and an augmented dataset of 151 cases. Each dataset was entered into UPM separately. The results created by UPM were reported by the automated procedure. A total of 3,528 final output files were created for each of the original and augmented data sets. Each file contains the detailed outcome of the experiments. Tables 2 and 3 report the summary of the results and the statistical data for the original dataset (132 cases) and the augmented dataset (151 cases), respectively.

Accuracy and Reliability of Prediction

From the results presented in Table 2, UPM's highest prediction rate is 90.90% on the original dataset, a marginally better prediction rate compared with the earlier studies obtained by Arditi et al. (1998), Arditi and Tokdemir (1999), and Arditi and Pulket (2005). UPM shows a marked improvement in the prediction rate when the augmented dataset (151 cases) is used. The highest prediction rate in this instance reached 96.02% (see Table 3). It is obvious that having more cases in the dataset enhances the performance of the model and, hence, increases the prediction rates.

Stratified tenfold cross-validation was used to measure the error. The data were divided randomly into 10 parts in which the class is represented in approximately the same proportions as in the full dataset. Each part was held out in turn and the remaining parts were trained. The error rate was calculated on the holdout set. The learning procedure was then executed a total of 10 times on different data sets. Finally, the 10 error estimates were averaged to yield an overall error estimate.

Prediction rates alone are usually inadequate to assess the performance of the model. A number of statistical parameters such as the kappa statistic, the mean square error (MSE), and the root-mean-square error (RMSE) were calculated to assess the reliability of UPM and to further guarantee its integrity.

The kappa statistic is an index of the degree of agreement between two or more classifiers classifying the same set of items, ranging from zero when agreement is no better than chance, to 1 when agreement is perfect. It is calculated by dividing the difference between the observed proportion of cases in which the classifiers agree and the proportion expected by chance by the maximum difference possible between the observed and expected proportions (Colman 2001). Landis and Koch (1977) have set a range for recommended kappa statistic values where 0 is poor, 0–0.20 slight, 0.21–0.40 fair, 0.41–0.60 moderate, 0.61–0.80 substantial, and 0.81–1 almost perfect strength of agreement.

The mean squared error (MSE) is the expected value of the square of the "error." The error is the amount by which the estimator differs from the quantity to be estimated. The difference occurs because of randomness or because the estimator does not account for information that could produce a more accurate estimate. The error is phrased as a sum of squares rather than the more intuitive sum of absolute errors because squares are differentiable across the entire real line, a key requirement of the method of least squares. The root-mean-square error (RMSE) is a

Table 3. Top 10 Prediction Rates of Augmented Data Set

Filenames	Prediction rate	Kappa statistic	RMSE	Total cases	Correct cases
2AB-010607.arff	96.0265	0.9422	0.1193	151	145
2AB-010609.arff	96.0265	0.9422	0.1196	151	145
2AB-010610.arff	96.0265	0.9422	0.1191	151	145
2AB-010627.arff	95.3642	0.9324	0.1202	151	144
2AB-010628.arff	95.3642	0.9324	0.1204	151	144
2AB-010604.arff	94.702	0.9225	0.1325	151	143
2AB-010613.arff	94.702	0.923	0.1299	151	143
2AB-010625.arff	94.702	0.923	0.1283	151	143
2AB-010640.arff	94.702	0.9225	0.1329	151	143
2AB-010642.arff	94.702	0.9225	0.1329	151	143

Table 4. Filename in Prediction Rate between Unaltered and Noisy Data Sets

Top 10 classifiers				
		Prediction rate (%)		
Filenames	Classifiers	Unaltered data set	Noise-added data set	Percent difference (%)
2AB-010609.arff	Boosting→J4.8	96.03	86.09	10.34
2AB-010610.arff	Boosting→JRip	96.03	82.12	14.48
2AB-010627.arff	Boosting→Part	96.03	86.76	9.66
2AB-010613.arff	MultiBoostAB→JRip	95.36	82.78	13.19
2AB-010625.arff	MultiBoostAB→Part	95.36	84.11	11.81
2AB-010607.arff	Original part	94.70	84.77	10.49
2AB-010628.arff	Bagging→J4.8	94.70	85.43	9.79
2AB-010604.arff	MultiBoostAB→J4.8	94.70	84.77	10.49
2AB-010640.arff	Grading comb 9	94.70	84.77	10.49
2AB-010642.arff	Grading comb 11	94.70	82.78	12.59
			Average	11.33

frequently used measure of the differences between values predicted by a model and the values actually observed from the thing being modeled or estimated. These individual differences are also called residuals, and the RMSE serves to aggregate them into a single measure of predictive power. Both MSE and RMSE should be close to zero if the error is minimal.

The kappa statistics and RMSEs of the top 10 results on both data sets are shown in Tables 2 and 3, indicating that UPM is quite reliable.

Robustness (Tolerance to Noisy Data)

Robustness is an indicator that shows the quality of the model being able to withstand with minimal damage or loss of functionality any alteration and abnormalities in input or changes in procedure or circumstances. In this study, robustness was measured by altering the original and augmented data sets by adding noise using a filter in the data consolidation process. Hence, the robustness of UPM was measured by comparing the prediction rates produced by unaltered data sets and noise-added data sets. By extension, the differences between the prediction rates were used to determine the robustness of individual hybrid classifiers. Classifiers that suffered a smaller difference in their prediction rate can withstand alterations and abnormalities, in this case, added noise, better than others, and, hence, were considered to be more robust.

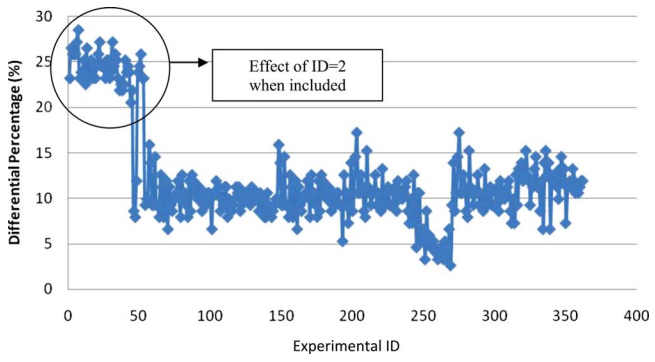

Fig. 4. Errors on noise-added data sets

The parameter in the "add noise" filter used on the augmented dataset was set to add 10% of noise to the dataset. Table 4 shows the best 10 prediction rates and shows that prediction rates suffered an average decline of 11.33%, only a marginally higher decline than the amount of noise actually added to the data. Note that the top 10 combinations are different in Tables 2 and 3 because Table 3 refers to the original dataset and Table 3 to the augmented dataset. Depending on the data sets used, UPM produces different solutions based on different combinations of tools.

The 362 experiments were tested to further study the effect of the added-noise filter to the data. The differences in prediction rates between unaltered and noisy data sets are plotted against experiment ID in Fig. 4. The classifiers that can withstand noise better result in lower percentage differences (%). When the plot is examined closely, it is observed that the experiment IDs from 2AB-020201 to 2AB-020264 spiked higher than the others. This is, however, not due to the classifiers but to the attribute selection process, CFS+BEST (ID=02) that causes erratic outcomes. If CFS+BEST is eliminated, the remaining results become reasonable with an average percentage error of 10.11%. Moreover, when CFS+BEST is eliminated, the standard deviation reduces from 5.43 to 2.97. There are some classifiers that are able to withstand abnormalities and still produce decent outcomes. These factors indicate that UPM performs well after all.

Computing Speed

In this section, the performance of UPM is reported relative to computational speed. The test was conducted by splitting the

Table 5. Computing Time Observed per Numbers of Experiment

ID of data set	Data consolidation method	Number of experiments	Computing time
1	1	588	16.29
2	1,2	1,176	33.49
3	1,2,3	1,764	50.17
4	1,2,3,4	2,352	68.5
5	1,2,3,4,5	2,940	86.03
6	1,2,3,4,5,6	3,528	103.13

```
J48 pruned tree

defendant 1 = own
|  non-excusable delay = y: con. (8.0/1.0)
|  non-excusable delay = n
|  |  type of plaintiff 1 = con
|  |  |  contract type = other
|  |  |  |  liquidated damages involved = n: own. (15.0)
|  |  |  |  liquidated damages involved = y
|  |  |  |  |  installation requirements = n: sub. (4.0/1.0)
|  |  |  |  |  installation requirements = y: own. (2.0/1.0)
|  |  |  contract type = fix
|  |  |  |  leincase involved = n: own. (10.0/2.0)
|  |  |  |  leincase involved = y
|  |  |  |  |  surety bonds = n: con. (13.0/2.0)
|  |  |  |  |  surety bonds = y: own. (5.0/2.0)
|  |  type of plaintiff 1 = supp: own. (0.0)
|  |  type of plaintiff 1 = own: own. (0.0)
|  |  type of plaintiff 1 = sub
|  |  |  esstoppel doctrine involved = n: sub. (8.0/2.0)
|  |  |  esstoppel doctrine involved = y: own. (8.0/2.0)
|  |  type of plaintiff 1 = other: own. (0.0)
|  |  type of plaintiff 1 = ae: ae. (2.0)
defendant 1 = con
|  contract type = other
|  |  non-excusable delay = y: own. (6.0/2.0)
|  |  non-excusable delay = n
|  |  |  type of plaintiff 1 = con: con. (0.0)
|  |  |  type of plaintiff 1 = supp
|  |  |  |  installation requirements = n: con. (2.0)
|  |  |  |  installation requirements = y: supp. (2.0)
|  |  |  type of plaintiff 1 = own: con. (12.0/1.0)
|  |  |  type of plaintiff 1 = sub: sub. (5.0)
|  |  |  type of plaintiff 1 = other: con. (5.0/1.0)
|  |  |  type of plaintiff 1 = ae: con. (0.0)
|  contract type = fix: own. (9.0)
defendant 1 = other: other. (8.0/2.0)
defendant 1 = sub: sub. (8.0/1.0)

Number of Leaves :  24
Size of the tree :  37
```

Fig. 5. General output format for decision trees

3,528 experiments into six smaller sets in the data consolidation process. The computing times were recorded to observe the effect of the number of experiments on computing time.

The data in Table 5 define a linear function of $y=33.72x+47.93$ that is almost perfectly aligned with the plot ($R^2=0.999$). It can, therefore, be seen that the computing time increases linearly relative to be number of data sets used. Hence, UPM can be expanded by adding more data consolidation methods and the anticipated computing time that the system consumes will be easily calculated since the relationship is quite predictable. Adding new attribute selection methods, base classifiers, meta learners, or ensemble methods is possible since the system is scalable and the additions of such tools will undoubtedly affect the computation time. The impact of such additions depends on the character of the tools added. In its current version at this stage, UPM did not show a run down on the computer's memory of 2 GBs on an Intel Core 2 Duo with a 2.13 GHz chip.

Interpretability

Interpretability in this study is meaningful because it can help with the understanding of the salient factors associated with construction claim cases in Illinois. As the classifiers themselves are categorized into many schemes such as decision trees, rule sets, and functions, the outputs always reflect the particular schemes. For example, in Fig. 5, the outputs are shown in decision tree format. This format is described as one of the best

```
JRIP rules:

(type of plaintiff 1 = supp) and (defendant 1 = con) and (installation requirements = y) => decision
=supp. (2.0/0.0)
(defendant 1 = other) => decision =other. (8.0/2.0)
(type of plaintiff 1 = ae) => decision =ae. (2.0/0.0)
(defendant 1 = sub) => decision =sub. (8.0/1.0)
(type of plaintiff 1 = sub) and (esstoppel doctrine involved = n) => decision =sub. (13.0/2.0)
(surety bonds = n) and (leincase involved = y) and (contract type = fix) and (type of plaintiff 1 =
con) => decision =con. (13.0/2.0)
(defendant 1 = con) and (non-excusable delay = n) and (contract type = other) => decision =con.
(19.0/2.0)
(non-excusable delay = y) and (type of plaintiff 1 = con) => decision =con. (8.0/1.0)
 => decision =own. (59.0/12.0)

Number of Rules : 9
```

Fig. 6. General output format for rule-based system

logical illustrations of a domain's patterns. The outputs of JRip in terms of rule sets are shown in Fig. 6. Unlike decision trees, rule sets give out the results in terms of a number of rules. This format also presents good logical reasoning. In Fig. 7, the output of a neural network system is presented. As observed in Fig. 7, neural network outputs are impossible to interpret since neural network modeling makes no logical attachment to the output. Therefore, neural networks can be less advantageous compared to other learning schemes in terms of interpretability.

Conclusion

Litigation occurs frequently in the construction industry. Preventing litigation can minimize time and cost consuming processes in the industry. In this study, a universal prediction model (UPM) was developed to predict the outcome of construction litigation more effectively than earlier models developed by Arditi et al. (1998), Arditi and Tokdemir (1999), and Arditi and Pulket (2005).

Earlier studies resulted in decent prediction rates. The ANN study resulted in a prediction rate of 66.67% (Arditi et al. 1998), the CBR study 83.33% (Arditi and Tokdemir 1999), and the BDT study 89.59% (Arditi and Pulket 2005). However, the earlier studies were composed of one algorithm, one experiment, and one result (Fig. 8). In contrast, UPM's approach (see Fig. 8) includes the use of a multitude of combinations of several data consolidation methods (six of them in this experiment, but could include all methods covered by WEKA), several attribute selection methods (12 in this experiment, but could be more de-

```
Sigmoid Node 14
Inputs    Weights
Threshold    -1.175016806343037
Attrib parties involved =privpub    -0.41567230920677534
Attrib parties involved =privpri    1.4649369933091663
Attrib parties involved =pubvpri    0.10304073738395836
Attrib defendant 1 =own    2.4909646688416576
Attrib defendant 1 =con    -2.841858879872153
Attrib defendant 1 =other    2.057055044605787
Attrib defendant 1 =sub    0.5616360107264305
Attrib resolution technique used    -0.8700718271132314
Attrib contract type    -2.6609283690921033
Attrib directed changes    0.7743170199863232
Attrib redical changes    -0.04505786773608095
Attrib unknown site conditions    -0.03480721552782278
Attrib CPM involved    0.04807912633621149
Attrib esstoppel doctrine involved    -1.6178479781204342
Attrib claim for material & equipment    -0.21043752915370317
Attrib installation requirements    0.8202274818081644
Attrib mispresentation of supervision    3.214144422561537
Attrib surety bonds    1.2363951893025902
Attrib leincase involved    1.9254120947164186
```

Fig. 7. General output format for neural network

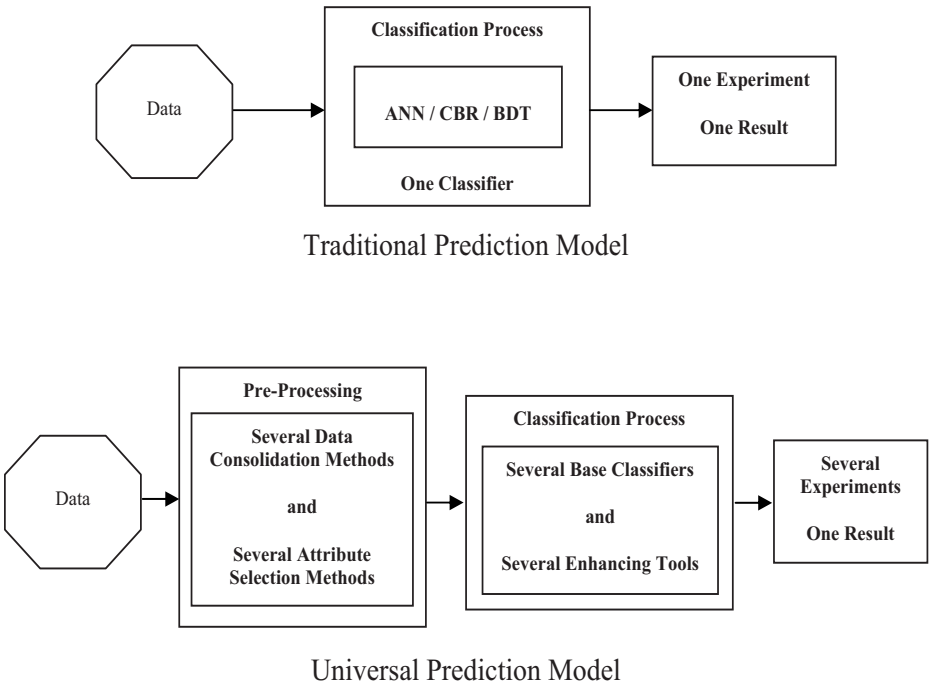

Fig. 8. Traditional approach in three previous studies

pending on the WEKA coverage), several base classifiers (only four out of the many covered by WEKA were picked for this experiment), and several enhancing tools (including three meta learners and three ensemble methods of the many listed in WEKA). The last part of UPM is the assessment of prediction results, relative to prediction accuracy, robustness, computing speed, and interpretability. What differentiates UPM from the traditional approach is its use of a combination of a multitude of methods in an organized, systematic, and automated way, hence, creating many outcomes (a total of 3,528 in this experiment), the best one of which is selected. UPM performed quite well in all four categories. It can be claimed that UPM is applicable not only to the construction litigation domain, but also, with minor adjustments, to any domain.

References

AbouRizk, S. M., and Dozzi, S. P. (1993). "Application of computer simulation in resolving construction disputes." *J. Constr. Eng. Manage.*, 119(2), 355–373.

Arditi, D., Oksay, F. E., and Tokdemir, O. B. (1998). "Predicting the outcome of construction litigation using neural networks." *Comput. Aided Civ. Infrastruct. Eng.*, 13(2), 75–81.

Arditi, D., and Pulket, T. (2005). "Predicting the outcome of construction litigation using boosted decision trees." *J. Comput. Civ. Eng.*, 19(4), 387–393.

Arditi, D., and Tokdemir, O. B. (1999). "A comparison of case-based reasoning and artificial neural networks." *J. Comput. Civ. Eng.*, 13(3), 162–169.

Bay, S. D., and Pazzani, M. J. (2000). "Characterizing model errors and differences." *Proc., 17th Int. Conf. on Machine Learning (ICML-2000)*, Morgan Kaufmann, San Francisco.

Cohen, W. W. (1995). "Fast effective rule induction." *Proc., 12th Int. Conf. on Machine Learning*, Morgan Kaufmann, San Mateo, Calif., 115–123.

Colman, A. M. (2001). *A dictionary of psychology*, Oxford University Press, Oxford, U.K.

Diekmann, J. E., and Girard, M. J. (1995). "Are contract disputes predictable?" *J. Constr. Eng. Manage.*, 121(4), 355–363.

Elazouni, A. M. (2006). "Classifying construction contractors using unsupervised-learning neural networks." *J. Constr. Eng. Manage.*, 132(12), 1242–1253.

Frank, E., and Witten, I. H. (1998). "Generating accurate rule sets without global optimization." *Proc., 15th Int. Conf.*, Morgan Kaufmann, San Francisco.

Gupta, R., Kewalramani, M. A., and Goel, A. (2006). "Prediction of concrete strength using neural-expert system." *J. Mater. Civ. Eng.*, 18(3), 462–466.

Han, J., and Kamber, M. (2001). *Data mining concepts and techniques*, Morgan Kaufmann, San Francisco, Calif.

Helmer, J. (2005). "Directory file listing utility for Microsoft Excel." Financial Reporting, Inc., ⟨www.finrepinc.com⟩ (Dec. 18, 2007).

Kassab, M., Hipel, K., and Hegazy, T. (2006). "Conflict resolution in construction disputes using the graph model." *J. Constr. Eng. Manage.*, 132(10), 1043–1052.

Kilian, J. J., and Gibson, G. E. (2005). "Construction litigation for the U.S. Naval Facilities engineering command, 1982–2002." *J. Constr. Eng. Manage.*, 131(9), 945–952.

Kotsiantis, S. B., and Pintelas, P. E. (2004). "Combining bagging and boosting." *Int J. Computational Intelligence*, 1(4), 324–333.

Kuncheva, L. I. (2004). *Combining pattern classifiers: Methods and algorithms*, Wiley, New York.

Landis, J. R., and Koch, G. G. (1977). "The measurement of observer agreement for categorical data." *Biometrics*, 33(1), 159–174.

Nashvili, M. (2006). "Decision trees and data mining. Natural computation." Dept. of Computer Science, Univ. of Birmingham, U.K., ⟨http://decisiontrees.net/node/36⟩ (Dec. 18, 2007).

Pena-Mora, F., Sosa, C. E., and McCone, D. S. (2003). *Introduction to construction dispute resolution*, MIT–Prentice-Hall Series, Upper Saddle River, N.J.

Quinlan, R. (1993). *C4.5: Programs for machine learning*, Morgan Kaufmann, San Mateo, Calif.

Ren, Z., Anumba, C. J., and Ugwu, O. O. (2003). "Multiagent system for construction claims negotiation." *J. Comput. Civ. Eng.*, 17(3), 180–188.

Russell, J. S. (1996). "Predicting contractor failure using stochastic dynamics of economic and financial variables." *J. Constr. Eng. Manage.*, 122(2), 183–191.

Seewald, A. K. (2002). "How to make stacking better and faster while also taking care of an unknown weakness." *Proc., 19th Int. Conf. on Machine Learning*, Morgan Kaufmann, San Francisco.

Seewald, A. K., and Fürnkranz, J. (2001). "Grading classifiers." *Proc., 2001 Intelligent Data Analysis Conf.*, extended version.

Webb, G. I. (2000). "MultiBoosting: A technique for combining boosting and wagging." *Mach. Learn.*, 40, 159–197.

Witten, I. H., and Frank, E. (2005). *Data mining, practical machine learning tools and techniques*, Morgan Kaufmann, San Francisco.

Wolpert, D. H. (1992). "Stacked generalization." *Neural Networks*, 5(2), 241–259.

Yiu, T. W., Cheung, S. O., and Mok, F. M. (2006). "Logistic likelihood analysis of mediation outcomes." *J. Constr. Eng. Manage.*, 132(10), 1026–1036.