Automated Design of User Profiling Systems for Fraud Detection

From: AAAI Technical Report WS-98-07. Compilation copyright © 1998, AAAI (www.aaai.org). All rights reserved.

Tom Fawcett and **Foster Provost**

Bell Atlantic Science and Technology
400 Westchester Avenue
White Plains, New York 10604
{fawcett,foster}@basit.com

Abstract

One method for detecting fraud is to check for suspicious changes in user behavior over time. This paper describes the automatic design of user profiling methods for the purpose of fraud detection, using a series of data mining and machine learning techniques. It uses a rule-learning program to uncover indicators of fraudulent behavior from a large database customer transactions. Then the indicators are used to create a set of monitors, which profile legitimate customer behavior and indicate anomalies. Finally, the outputs of the monitors are used as features in a system that learns to combine evidence to generate high-confidence alarms. The system has been applied to the problem of detecting cellular cloning, but is applicable to a more general class of fraud called superimposition fraud. Experiments indicate that this automatic approach performs better than hand-crafted methods for detecting fraud. Furthermore, this approach can adapt to the changing conditions typical of fraud detection environments.

Introduction

In the United States, cellular fraud costs the telecommunications industry hundreds of millions of dollars per year (Walters & Wilkinson 1994; Steward 1997). One kind of cellular fraud called *cloning* is particularly expensive and epidemic in major cities throughout the United States. Cloning fraud causes great inconvenience to customers and great expense to cellular service providers. Existing methods for detecting cloning fraud are *ad hoc* and their evaluation is virtually nonexistent. We have embarked on a program of systematic analysis of cellular call data for the purpose of designing and evaluating methods for detecting fraudulent behavior.

Cloning fraud is one instance of *superimposition fraud*, in which fraudulent usage is superimposed upon (added to) the legitimate usage of an account. Other examples are credit card fraud, calling card fraud and some forms of computer intrusion. Superimposition fraud typically occurs when a non-legitimate user gains illicit access to the account or service of a legitimate user. Superimposition fraud is detectable if the legitimate users have fairly regular behavior that is generally distinguishable from the fraudulent behavior.

Our framework includes a data mining component for discovering indicators of fraud. A constructive induction component generates profiling detectors that use the discovered indicators. A final evidence-combining component determines how to combine signals from the profiling detectors to generate alarms. The rest of this paper describes the domain, the framework and the implemented system, the data, and results.

Cellular Cloning Fraud and its Detection

Every cellular phone periodically transmits two unique identification numbers: its *Mobile Identification Number* (MIN) and its *Electronic Serial Number* (ESN). These two numbers are broadcast unencrypted over the airwaves, and can be received, decoded and stored using special equipment that is relatively inexpensive. *Cloning* occurs when a customer's MIN and ESN are programmed into a cellular telephone not belonging to the customer. When this telephone is used, the network sees the customer's MIN and ESN and subsequently bills the usage to the customer. With the stolen MIN and ESN, a cloned phone user (whom we shall call a *bandit*) can make virtually unlimited calls, whose charges are billed to the customer.[1] If the fraudulent usage goes undetected, the customer's next bill will include the corresponding charges. Typically, the customer then calls the cellular service provider (the *carrier*) and denies the usage. The carrier and cus-

Copyright ©1998, American Association for Artificial Intelligence (www.aaai.org). All rights reserved. An earlier version of this paper appeared in *Proceedings of The Second Int'l Conf. on Knowledge Discovery and Data Mining.* Menlo Park, CA, pp. 8–13. AAAI Press. 1996.

[1] According to the Cellular Telecommunications Industry Association, MIN-ESN pairs are sold on the streets of major US cities for between $5 and $50 apiece.

tomer then determine which calls were made by the "bandit" and which were legitimate calls. The fraudulent charges are credited to the customer's account, and measures are taken to prohibit further fraudulent charges, usually by assigning the customer a (new) Personal Identification Number.

Fraud causes considerable inconvenience both to the carrier and to the customer. Fraudulent usage also incurs significant financial losses due to costs of land-line usage (most cellular calls are to non-cellular destinations), costs of congestion in the cellular system, loss of revenue by the crediting process, and costs paid to other cellular companies when a customer's MIN and ESN are used outside the carrier's home territory.

Cellular carriers therefore have a strong interest in detecting cloning fraud as soon as possible. Standard methods of fraud detection include analyzing call data for overlapping calls (*collisions*), or calls in temporal proximity that could not have been placed by the same user due to geographic dispersion (*velocity checks*) (Davis & Goyal 1993). More sophisticated methods involve profiling user behavior and looking for significant deviations from normal patterns. This paper addresses the automatic design of such methods.

One approach to detecting fraud automatically is to learn a classifier for individual calls. We have not had success using standard machine learning techniques to construct such a classifier. Context is very important: a call that would be unusual for one customer would be typical for another. Furthermore, legitimate subscribers occasionally make isolated calls that look suspicious, so in general decisions of fraud should not be made on the basis of individual calls (Fawcett & Provost 1997).

To detect fraud reliably it is necessary to determine the normal behavior of each account with respect to certain indicators, and to determine when that behavior has deviated significantly. Three issues arise:

1. *Which call features are important?* Which features or combinations of features are useful for distinguishing legitimate behavior from fraudulent behavior?

2. *How should profiles be created?* Given an important feature identified in Step 1, how should we characterize the behavior of a subscriber with respect to the feature?

3. *When should alarms be issued?* Given a set of profiling criteria identified in Step 2, how should we combine them to determine when fraud has occurred?

Our goal is to automate the design of user-profiling systems. Each of these issues corresponds to a component of our framework.

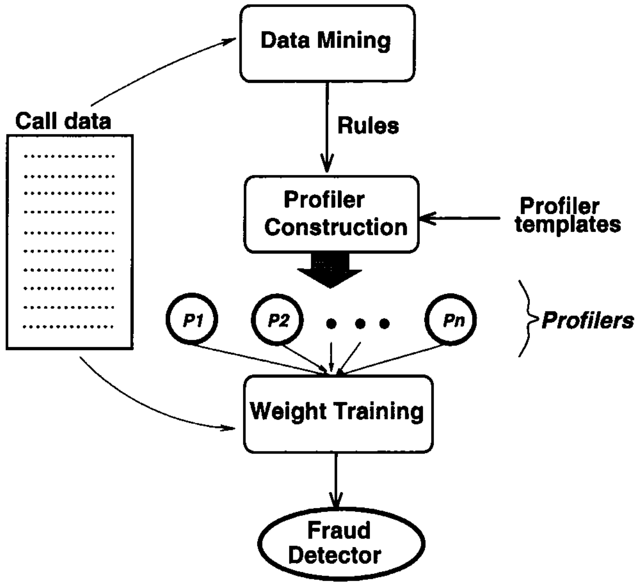

Figure 1: A framework for automatically constructing fraud detectors.

The Framework and the DC-1 System

Our system framework is illustrated in Figure 1. The framework uses data mining to discover indicators of fraudulent behavior, and then builds modules to profile each user's behavior with respect to these indicators. The *profilers* capture the typical behavior of an account and, in use, describe how far an account is from this typical behavior. The profilers are combined into a single *detector*, which learns how to detect fraud effectively based on the profiler outputs. When the detector has enough evidence of fraudulent activity on an account, based on the indications of the profilers, it generates an alarm.

Figure 1 depicts the automatic generation of a fraud detector from a set of data on fraudulent and legitimate calls. The system takes as input a set of *call data*, which are chronological records of the calls made by each subscriber, organized by account. The call data describe individual calls using features such as TIME-OF-DAY, DURATION and CELL-SITE. The constructor also takes as input a set of *profiler templates*, which are the basis for the construction of the individual profilers.

Mining the Call Data

The first stage of detector construction, *data mining*, involves combing through the call data searching for indicators of fraud. In the DC-1 system, the indicators are conjunctive rules discovered by a standard rule-learning program. We use the RL program (Clearwater & Provost 1990), which is similar to other Meta-DENDRAL-style rule learners (Buchanan & Mitchell 1978; Segal & Etzioni 1994). RL searches for rules with certainty factors above a user-defined threshold. The certainty factor we used for these runs was a simple frequency-based probability estimate, corrected for small samples (Quinlan 1987).

The call data are organized by account, and each call record is labeled as fraudulent or legitimate. When RL is applied to an account's calls it produces a set of rules that serve to distinguish, within that account, the fraudulent calls from the legitimate calls. As an example, the following rule would be a relatively good indicator of fraud:

```
(TIME-OF-DAY = NIGHT) AND (LOCATION = BRONX)
                  ==> FRAUD
          Certainty factor = 0.89
```

This rule denotes that a call placed at night from The Bronx (a Borough of New York City) is likely to be fraudulent. The `Certainty factor = 0.89` means that, for this account, a call matching this rule has an 89% probability of being fraudulent.

Each account generates a set of such rules. Each rule is recorded along with the account from which it was generated. After all accounts have been processed, a rule selection step is performed, the purpose of which is to derive a general covering set of rules that will serve as fraud indicators.

The set of accounts is traversed again. For each account, the list of rules generated by that account is sorted by the frequency of occurrence in the entire account set. The highest frequency unchosen rule is selected. If an account has been covered already by four chosen rules, it is skipped. The resulting set of rules is used in profiler construction.

Constructing Profilers

The second stage of detector construction, *profiler construction,* generates a set of profilers from the discovered fraud rules. The profiler constructor has a set of templates which are instantiated by rule conditions. The profiler constructor is given a set of rules and a set of templates, and generates a profiler from each rule-template pair. Every profiler has a *Training* step, in which it is trained on typical (non-fraud) account activity; and a *Use* step, in which it describes how far from the typical behavior a current account-day is. For example, a simple profiler template would be:

- **Given:** *Rule conditions* from a fraud rule.
- **Training:** On a daily basis, count the number of calls that satisfy *rule conditions.* Keep track of the maximum as *daily-threshold.*
- **Use:** Given an account-day, output 1 if the number of calls in a day exceeds *daily-threshold,* else output 0.

Assume the Bronx-at-night rule mentioned earlier was used with this template. The resulting instantiated profiler would determine, for a given account, the maximum number of calls made from The Bronx at night in any 24-hour period. In use, this profiler would emit a 1 whenever an account-day exceeded this threshold.

Different kinds of profilers are possible. A *thresholding* profiler yields a binary feature corresponding to whether the user's behavior was above threshold for the given day. A *counting* profiler yields a feature corresponding to its count (*e.g.*, the number of calls from BRONX at NIGHT). A *percentage* profiler yields a feature whose value is between zero and one hundred, representing the percentage of calls in the account-day that satisfy the conditions. Each type of profiler is produced by a different type of profiling template.

Combining Evidence from the Profilers

The third stage of detector construction learns how to combine evidence from the set of profilers generated by the previous stage. For this stage, the outputs of the profilers are used as features to a standard machine learning program. Training is done on account data, and profilers evaluate a complete account-day at a time. In training, the profilers' outputs are presented along with the desired output (the account-day's classification). The evidence combination learns which combinations of profiler outputs indicate fraud with high confidence.

Many training methods for evidence combining are possible. After experimenting with several methods, we chose a simple Linear Threshold Unit (LTU) for our experiments. An LTU is simple and fast, and enables a good first-order judgment of the features' worth.

A feature selection process is used to reduce the number of profilers in the final detector. Some of the rules do not perform well when used in profilers, and some profilers overlap in their fraud detection coverage. We therefore employ a sequential forward selection process (Kittler 1986) which chooses a small set of useful profilers. Empirically, this simplifies the final detector and increases its accuracy.

Account-Day

Day	Time	Duration	Origin	Destination
Tue	01:42	10 mins	Bronx, NY	Miami, FL
Tue	10:05	3 mins	Scrsdl, NY	Bayonne, NJ
Tue	11:23	24 sec	Scrsdl, NY	Congers, NY
Tue	14:53	5 mins	Trrytwn, NY	Grnwich,CT
Tue	15:06	5 mins	Manhat, NY	Wstport, CT
Tue	16:28	53 sec	Scrsdl, NY	Congers, NY
Tue	23:40	17 mins	Bronx, NY	Miami, FL

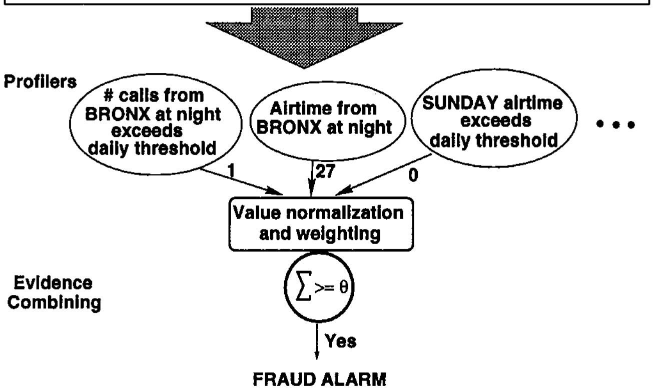

Figure 2: A DC-1 fraud detector processing a single account-day of data.

The Detector

The final output of the constructor is a detector that profiles each user's behavior based on several indicators, and produces an alarm if there is sufficient evidence of fraudulent activity. Figure 2 shows an example of a simple detector evaluating an account-day.

Before being used on an account, the profilers undergo a *profiling period* (usually 30 days) during which they measure unfrauded usage. In our study, these initial 30 account-days were guaranteed free of fraud, but were not otherwise guaranteed to be typical. From this initial profiling period, each profiler measures a characteristic level of activity.

The Data

The call data used for this study are records of cellular calls placed over four months by users in the New York City area—an area with high levels of fraud. Each call is described by thirty-one attributes, such as the phone number of the caller, the duration of the call, the geographical origin and destination of the call, and any long-distance carrier used. Because of security considerations, we are unable to disclose all the features used in the system.

To these thirty-one attributes are added several derived attributes that incorporate knowledge we judged to be potentially useful. One such attribute is a categorical TIME-OF-DAY variable representing the time segment of the day in which a call is placed. Its values are MORNING, AFTERNOON, TWILIGHT, EVENING and NIGHT. Another derived attribute is TO-PAYPHONE, a binary flag indicating whether the call terminated at a payphone. Note that any number of additional features could be added to encode relevant domain knowledge.

Each call is also give a class label of legitimate or fraudulent. This is done by cross referencing a database of all calls that were credited as being fraudulent for the same time period.

Rule learning and selection used 879 accounts comprising over 500,000 calls. About 3600 accounts were selected for profiling, training, and testing. The only condition used to select these 3600 accounts was that they be guaranteed to have at least thirty fraud-free days of usage before any fraudulent usage. The initial thirty days of each account were used for profiling. The remaining days of usage were used to generate approximately 96,000 account-days. Using randomly selected accounts, we generated sets of 10,000 account-days for training and 5000 account-days for testing. Training and testing accounts were distinct, so their account-days were not mixed between training and testing.[2] Each set of account-days was chosen to comprise 20% fraud and 80% non-fraud days.

Results

Data mining generated 3630 rules, each of which applied to two or more accounts. The rule selection process, in which rules are chosen in order of maximum account coverage, yielded a smaller set of 99 rules sufficient to cover the accounts. Each of the 99 rules was used to instantiate two profiler templates, yielding 198 profilers. The final feature selection step reduced this to nine profilers, with which the experiments were performed.

Each detector was run ten times on randomly selected training and testing accounts. Accuracy averages and standard deviations are shown in the leftmost column of Table 1. For comparison, we evaluated DC-1 along with other detection strategies:

- **Alarm on All** represents the policy of alarming on every account every day. The opposite strategy, **Alarm on None**, represents the policy of allowing

[2] If account-days from a single account appear in both training and testing sets, the performance evaluation can be deceptively optimistic. Fraudulent behavior within a specific cloning episode is more similar than fraudulent behavior between episodes. When deployed, the monitors will be used to search for previously unseen cloning episodes.

Table 1: Accuracies and costs of various detectors.

Detector	Accuracy (%)	Cost (US$)	Accuracy at cost (%)
Alarm on All	20	20000	20
Alarm on None	80	18111 ± 961	80
Collisions + Velocities	82 ± .3	17578 ± 749	82 ± .4
High Usage	88 ± .7	6938 ± 470	85 ± 1.7
Best individual DC-1 monitor	89 ± .5	7940 ± 313	85 ± .8
State of the Art (SOTA)	90 ± .4	6557 ± 541	88 ± .9
DC-1 detector	92 ± .5	5403 ± 507	91 ± .8
SOTA plus DC1	92 ± .4	5078 ± 319	91 ± .8

fraud to go completely unchecked. The latter corresponds to the maximum likelihood accuracy classification. Note that the cost of **Alarm on None** does not take into account the inhibitory effect of fraud detection, without which fraud levels would likely continue to rise.

- **Collisions and Velocities** is a detector using collision and velocity checks, described earlier. DC-1 was used to learn a threshold on the number of collision and velocity alarms necessary to generate a fraud alarm. It is surprising that Collisions and Velocity Checks, commonly thought to be reliable indicators of cloning, performed poorly in our experiments.

  The performance of collisions and velocity checks was originally worse than reported here because of false alarms. Manual inspection of false alarms revealed a few synchronization problems; for example, some apparent collisions were caused when a call was dropped then quickly re-established in a neighboring cell whose clock did not agree with the first cell's. Some such conditions could be caught easily, so we patched the detection algorithms to check for them. The results in this paper are for the improved detectors.

  Investigation of confusion matrices revealed that the collision and velocity check detectors' errors were due almost entirely to false negatives. In other words, when the detectors fired they were accurate, but many fraud days never exhibited a collision or velocity check.

- Some fraud analysts believe that cloning fraud is usually accompanied by large jumps in account usage, and sophisticated mining of fraud indicators is probably unnecessary since most fraud could be caught by looking for sudden increases in usage. We created the **High Usage** detector to test this hypothesis. It generates alarms based only on amount of usage. It is essentially a standard deviation monitor (Fawcett & Provost 1997) whose rule conditions are always satisfied. The threshold of this detector was found empirically from training data.

  Note that the evaluation of cost for the high usage detector may be overly optimistic, due to inadequacies in our cost model. In particular, a trained high usage detector learns to optimally "skim the cream," without regard to the fact that the errors it makes will involve annoying the best customers. In these cases, the cost of a false alarm may be much higher than the fixed cost we assigned.

- The **Best Individual DC-1 Monitor** was used as an isolated detector. This experiment was done to determine the additional benefit of combining monitors. The best individual monitor was generated from the rule:

  ```
  (TIME-OF-DAY = EVENING) ==> FRAUD
  ```

  Rule learning had discovered (in 119 accounts) that the sudden appearance of evening calls, in accounts that did not normally make them, was coincident with cloning fraud. The relatively high accuracy of this one monitor reveals that this is a valuable fraud indicator.

  Our `TIME-OF-DAY` attribute has five possible values: MORNING, AFTERNOON, TWILIGHT, EVENING and NIGHT. Although EVENING is by far the most frequent value implicated in fraud, rule learning generated fraud rules involving each of these values. This suggests that any time-of-day change in a subscriber's normal behavior may be indicative of fraud, though the other shifts may not be predictive enough to use in a fraud monitor.

- The **DC-1** detector incorporates all the monitors chosen by feature selection. We used the weight learning method described earlier to determine the weights for evidence combining.

- The **SOTA** ("State Of The Art") detector incorporates thirteen hand-crafted profiling methods that were the best individual detectors identified in a previous study. Each method profiles an account in a different way and produces a separate alarm. Weights for combining SOTA's alarms were determined by our weight-tuning algorithm.

In this domain, different types of errors have different costs, and a realistic evaluation must take these costs into account. A false positive error (a false alarm) corresponds to wrongly deciding that a customer has been cloned. Based on the cost of a fraud analyst's time, we estimate the cost of a false positive error to be about \$5. A false negative error corresponds to letting a frauded account-day go undetected. Rather than using a uniform cost for all false negatives, we estimated a false negative to cost \$.40 per minute of fraudulent airtime used on that account-day. This figure is based on the proportion of usage in local and non-local ("roaming") markets, and their corresponding costs.[3]

Because LTU training methods try to minimize errors but not error costs, we employed a second step in training. After training, the LTU's threshold is adjusted to yield minimum error cost on the training set. This adjustment is done by moving the decision threshold from -1 to +1 in increments of .01 and computing the resulting error cost. After the minimum cost on training data is found, the threshold is clamped and the testing data are evaluated. The second column of Table 1 shows the mean and standard deviations of test set costs. The third column, "Accuracy at cost," is the corresponding classification accuracy of the detector when the threshold is set to yield lowest-cost classifications.

Discussion

The results in Table 1 demonstrate that DC-1 performs quite well. In fact, DC-1 outperforms SOTA in terms of both accuracy and cost.[4] In our experiments, lowest cost classification occurred at an accuracy somewhat lower than optimal. In other words, some classification accuracy can be sacrificed to decrease cost. More sophisticated methods could be used to produce cost sensitive classifiers, which would probably produce better results.

Finally, the monitors of SOTA and DC-1 were combined into a hybrid detector. The resulting detector (**SOTA plus DC-1**) exhibits no increase in classification accuracy, but does show a slight improvement in fraud detection cost.

Related Work

Yuhas (1993) and Ezawa and Norton (1995) address the problem of uncollectible debt in telecommunications services. However, neither work deals with characterizing typical customer behavior, so mining the data to derive profiling features is not necessary. Ezawa and Norton's method of evidence combining is much more sophisticated than ours and faces some of the same problems (unequal error costs, skewed class distributions).

Methods that deal with time series are relevant to our work. However, most time series analysis (Chatfield 1984; Farnum & Stanton 1989) strives to characterize an entire time series or to forecast future events in the series. Neither ability is directly useful to fraud detection. Hidden Markov Models (Rabiner & Juang 1986) are concerned with distinguishing recurring sequences of states and the transitions between them. However, fraud detection usually only deals with two states (the "frauded" and "un-frauded" states) with a single transition between them. It may be useful to recognize recurring un-frauded states of an account, but this ability is likely peripheral to the detection task.

A longer article based on this work (Fawcett & Provost 1997) evaluates the performance of DC-1 on shifting distributions of fraud, as well as the use of fraudulent call classifiers.

Conclusions and Future Work

It is difficult to evaluate DC-1 against existing expert systems for fraud detection. Fraud detection departments carefully protect information about how much fraud they have and how effective their detection strategies are. Likewise, vendors of fraud detection systems protect details of their systems' operation that may constitute trade secrets. Little performance data on fielded systems are available, and what data do exist are insufficient for careful evaluation.

For these reasons, we evaluated DC-1 against individual known fraud detection techniques, as well as against a collection of techniques representing the state of the art as we understand it. Results in the previous sections show that the DC-1 detector performs better than the high-usage alarm and the collision/velocity

[3]We have still glossed over some complexity. For a given account, the only false negative fraud days that incur cost to the company are those prior to the *first* true positive alarm. After the fraud is detected, it is terminated. Thus, our analysis overestimates the costs slightly; a more thorough analysis would eliminate such days from the computation.

[4]Earlier work (Fawcett & Provost 1996) reported a higher accuracy for SOTA than is shown here. Further development of SOTA revealed that some of its component methods, developed in a prior study, had been built from account data that overlapped data used to test the methods. When a strict separation was enforced, SOTA performance declined slightly to the figures shown here.

alarm. DC-1 also out-performs the SOTA detector, consisting of a collection of the best fraud detection techniques known to us, trained by DC-1's evidence combining method.

DC-1's framework has three main components, and is more complex than other approaches. Our experiments were designed not only to evaluate the overall performance of the system, but also to analyze the contribution of the individual components. In particular:

- The High Usage detector profiles with respect to undifferentiated account usage. Comparison with DC-1's performance demonstrates the benefit of using rule learning to uncover specific indicators of fraudulent calls.
- The Call Classifier detectors represent rule learning without the benefit of account context. Comparison with DC-1's performance demonstrates the value of DC-1's rule generation step, which does preserve account context.
- Comparison of DC-1 with the single best individual DC-1 monitor demonstrates the benefit of combining evidence from multiple monitors.
- Experiments with shifting fraud distributions (Fawcett & Provost 1997) indicate the benefit of making evidence combination sensitive to fraud distributions.

In each of these cases, the composite DC-1 system out-performed the detector in which a significant piece was missing. These results suggest that each component contributes critically to the performance of the entire detector.

We believe our framework will be useful in other domains in which typical behavior is to be distinguished from unusual behavior. Prime candidates are similar domains involving fraud, such as credit-card fraud and toll fraud. In credit-card fraud, data mining may identify locations that arise as new hot-beds of fraud. The constructor would then incorporate profilers that notice if a customer begins to charge more than usual from that location.

Acknowledgements

We would like to thank Nicholas Arcuri and the Fraud Control department at Bell Atlantic NYNEX Mobile for many useful discussions about cellular fraud and its detection.

References

Buchanan, B. G., and Mitchell, T. M. 1978. Model-directed learning of production rules. In Hayes-Roth, F., ed., *Pattern-directed inference systems.* New York: Academic Press. 297–312.

Chatfield, C. 1984. *The analysis of time series: An introduction (third edition).* New York: Chapman and Hall.

Clearwater, S., and Provost, F. 1990. RL4: A tool for knowledge-based induction. In *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, 24–30. IEEE CS Press.

Davis, A., and Goyal, S. 1993. Management of cellular fraud: Knowledge-based detection, classification and prevention. In *Thirteenth International Conference on Artificial Intelligence, Expert Systems and Natural Language*, volume 2, 155–164.

Ezawa, K., and Norton, S. 1995. Knowledge discovery in telecommunication services data using bayesian network models. In Fayyad, U., and Uthurusamy, R., eds., *Proceedings of First International Conference on Knowledge Discovery and Data Mining*, 100–105. Menlo Park, CA: AAAI Press.

Farnum, N., and Stanton, L. 1989. *Quantitative forecasting methods.* Boston, MA: PWS-Kent Publishing Company.

Fawcett, T., and Provost, F. 1996. Combining data mining and machine learning for effective user profiling. In Simoudis, E.; Han, J.; and Fayyad, U., eds., *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 8–13. Menlo Park, CA: AAAI Press.

Fawcett, T., and Provost, F. 1997. Adaptive fraud detection. *Data Mining and Knowledge Discovery* 1(3). Available: `http://www.croftj.net/~fawcett/DMKD-97.ps.gz`.

Kittler, J. 1986. Feature selection and extraction. In Fu, K. S., ed., *Handbook of pattern recognition and image processing.* New York: Academic Press. 59–83.

Quinlan, J. R. 1987. Generating production rules from decision trees. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 304–307. Morgan Kaufmann.

Rabiner, L. R., and Juang, B. H. 1986. An introduction to hidden markov models. *IEEE ASSB Magazine* 3(1):4–16.

Segal, R., and Etzioni, O. 1994. Learning decision lists using homogeneous rules. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 619–625. Menlo Park, CA: AAAI Press.

Steward, S. 1997. Lighting the way in '97. *Cellular Business* 23.

Walters, D., and Wilkinson, W. 1994. Wireless fraud, now and in the future: A view of the problem and some solutions. *Mobile Phone News* 4–7.

Yuhas, B. P. 1993. Toll-fraud detection. In Alspector, J.; Goodman, R.; and Brown, T., eds., *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications*, 239–244. Hillsdale, NJ: Lawrence Erlbaum Associates.