To Buy or Not to Buy: Mining Airfare Data to Minimize Ticket Purchase Price

Oren Etzioni
Dept. Computer Science
University of Washington
Seattle, Washington 98195
etzioni@cs.washington.edu

Craig A. Knoblock
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
knoblock@isi.edu

Rattapoom Tuchinda
Dept. of Computer Science
University of Southern California
Los Angeles, CA 90089
pipet@isi.edu

Alexander Yates
Dept. Computer Science
University of Washington
Seattle, Washington 98195
ayates@cs.washington.edu

ABSTRACT

As product prices become increasingly available on the World Wide Web, consumers attempt to understand how corporations vary these prices over time. However, corporations change prices based on proprietary algorithms and hidden variables (e.g., the number of unsold seats on a flight). Is it possible to develop data mining techniques that will enable consumers to predict price changes under these conditions?

This paper reports on a pilot study in the domain of airline ticket prices where we recorded over 12,000 price observations over a 41 day period. When trained on this data, Hamlet — our multi-strategy data mining algorithm — generated a predictive model that saved 341 simulated passengers $198,074 by advising them when to buy and when to postpone ticket purchases. Remarkably, a clairvoyant algorithm with complete knowledge of future prices could save at most $320,572 in our simulation, thus HAMLET's savings were 61.8% of optimal. The algorithm's savings of $198,074 represents an average savings of 23.8% for the 341 passengers for whom savings are possible. Overall, HAMLET saved 4.4% of the ticket price averaged over the entire set of 4,488 simulated passengers. Our pilot study suggests that mining of price data available over the web has the potential to save consumers substantial sums of money per annum.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

SIGKDD '03, August 24-27, 2003, Washington, DC, USA.

Keywords

price mining, Internet, web mining, airline price prediction

1. INTRODUCTION AND MOTIVATION

Corporations often use complex policies to vary product prices over time. The airline industry is one of the most sophisticated in its use of dynamic pricing strategies in an attempt to maximize its revenue. Airlines have many fare classes for seats on the same flight, use different sales channels (e.g., travel agents, priceline.com, consolidators), and frequently vary the price per seat over time based on a slew of factors including seasonality, availability of seats, competitive moves by other airlines, and more. The airlines are said to use proprietary software to compute ticket prices on any given day, but the algorithms used are jealously guarded trade secrets [19]. Hotels, rental car agencies, and other vendors with a "standing" inventory are increasingly using similar techniques.

As product prices become increasingly available on the World Wide Web, consumers have the opportunity to become more sophisticated shoppers. They are able to comparison shop efficiently and to track prices over time; they can attempt to identify pricing patterns and rush or delay purchases based on anticipated price changes (e.g., "I'll wait to buy because they always have a big sale in the spring..."). In this paper we describe the use of data mining methods to help consumers with this task. We report on a pilot study in the domain of airfares where an automatically learned model, based on price information available on the Web, was able to save consumers a substantial sum of money in simulation.

The paper addresses the following central questions:

- **What is the behavior of airline ticket prices over time?** Do airfares change frequently? Do they move in small increments or in large jumps? Do they tend to go up or down over time? Our pilot study enables us to begin to characterize the complex behavior of airfares.

- **What data mining methods are able to detect patterns in price data?** In this paper we consider reinforcement learning, rule learning, time series methods, and combinations of the above.

- **Can Web price tracking coupled with data mining save consumers money in practice?** Vendors vary prices based on numerous variables whose values are not available on the Web. For example, an airline may discount seats on a flight if the number of unsold seats, on a particular date, is high relative to the airline's model. However, consumers do not have access to the airline's model or to the number of available seats on the flights. Thus, *a priori*, price changes could appear to be unpredictable to a consumer tracking prices over the Web. In fact, we have found price changes to be surprisingly predictable in some cases.

The remainder of this paper is organized as follows. Section 2 describes our data collection mechanism and analyzes the basic characteristics of airline pricing in our data. Section 3 considers related work in the areas of computational finance and time series analysis. Section 4 introduces our data mining methods and describes how each method was tailored to our domain. We investigated rule learning [8], Q-learning [25], moving average models [13], and the combination of these methods via stacked generalization [28]. Next, Section 5 describes our simulation and the performance of each of the methods on our test data. The section also reports on a sensitivity analysis to assess the robustness of our results to changes in the simulation. We conclude with a discussion of future work and a summary of the paper's contributions.

2. DATA COLLECTION

We collected airfare data directly from a major travel web site. In order to extract the large amount of data required for our machine learning algorithms, we built a flight data collection agent that runs at a scheduled interval, extracts the pricing data, and stores the result in a database.

We built our flight data collection agent using AgentBuilder[1] for wrapping web sites and Theseus for executing the agent [3]. AgentBuilder exploits machine learning technology [15] that enables the system to automatically learn extraction rules that reliably convert information presented on web pages into XML. Once the system has learned the extraction rules, AgentBuilder compiles this into a Theseus plan. Theseus is a streaming dataflow execution system that supports highly optimized execution of a plan in a network environment. The system maximizes the parallelism across different operators and streams data between operations to support the efficient execution of plans with complex navigation paths and extraction from multiple pages.

For the purpose of our pilot study, we restricted ourselves to collecting data on non-stop, round-trip flights for two routes: Los Angeles (LAX) to Boston (BOS) and Seattle (SEA) to Washington, DC (IAD). Our departure dates spanned January 2003 with the return flight 7 days after departure. For each departure date, we began collecting pricing data 21 days in advance at three-hour intervals; data for each departure date was collected 8 times a day.[2] Overall, we collected over 12,000 fare observations over a 41 day period for six different airlines including American, United, etc. We used three-hour intervals to limit the number of `http` requests to the web site. For each flight, we recorded the *lowest* fare available for an *economy* ticket. We also recorded when economy tickets were no longer available; we refer to such flights as *sold out*.

2.1 Pricing Behavior in Our Data

We found that the price of tickets on a particular flight can change as often as seven times in a single day. We categorize price change into two types: dependent price changes and independent price changes. Dependent changes occur when prices of similar flights (i.e. having the same origin and destination) from the same airline change at the same time. This type of change can happen as often as once or twice a day when airlines adjust their prices to maximize their overall revenue or "yield". Independent changes occur when the price of a particular flight changes independently of similar flights from the same airline. We speculate that this type of change results from the change in the seat availability of the particular flight. Table 1 shows the average number of changes per flight aggregated over all airlines for each route. Overall, 762 price changes occurred across all the flights in our data. 63% of the changes can be classified as dependent changes based on the behavior of other flights by the same airline.

Route	*Avg. number of price changes*
LAX-BOS	6.8
SEA-IAD	5.4

Table 1: Average number of price changes per route.

We found that the round-trip ticket price for flights can vary significantly over time. Table 2 shows the minimum price, maximum price, and the maximum difference in prices that can occur for flights on each route.

Route	*Min Price*	*Max Price*	*Max Price Change*
LAX-BOS	275	2524	2249
SEA-IAD	281	1668	1387

Table 2: Minimum price, maximum price, and maximum change in ticket price per route. All prices in this paper refer to the lowest economy airfare available for purchase.

For many flights there are easily discernible price tiers where ticket prices fall into a relatively small price range. The number of tiers typically varies from two to four, depending on the airline and the particular flight. Even flights from the same airline with the same schedule (but with different departure dates) can have different numbers of tiers. For example, there are two price tiers for the flight in Figure 1, four price tiers in Figure 4 and three price tiers in Figure 2 and Figure 3.

[1] `www.fetch.com`

[2] We expected to record 168 $(21 * 8)$ price observations for each flight. In fact, we found that on average each flight was missing 25 observations due to problems during data collection including remote server failures, site changes, wrapper bugs, etc.

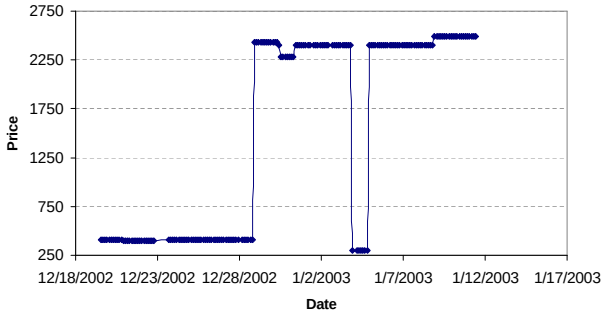

Figure 1: Price change over time for United Airlines roundtrip flight#168:169 LAX-BOS departing on Jan 12. This figure is an example of two price tiers and how consumers might benefit from the price drop.

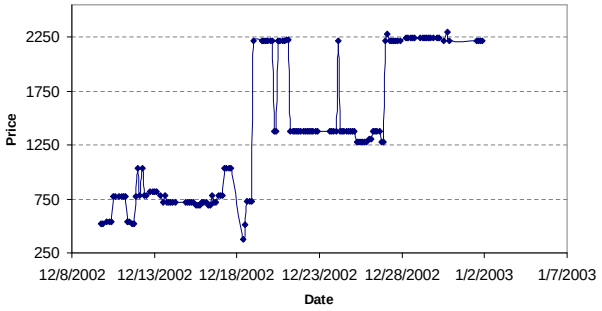

Figure 2: Price change over time for American Airlines roundtrip flight#192:223, LAX-BOS departing on Jan 2. This figure shows an example of rapid price fluctuation in the days priori to the New Year.

Price matching plays an important role in airline pricing structure. Airlines use sophisticated software to track their competitors' pricing history and propose adjustments that optimize their overall revenue. To change the price, airlines

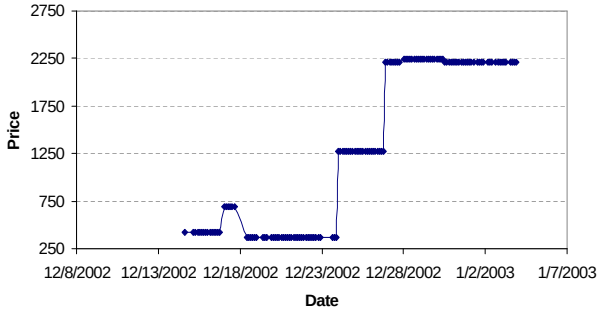

Figure 3: Price change over time for American Airlines roundtrip flight#192:223, LAX-BOS departing on Jan 7. This figure shows an example of three price tiers and low price fluctuation

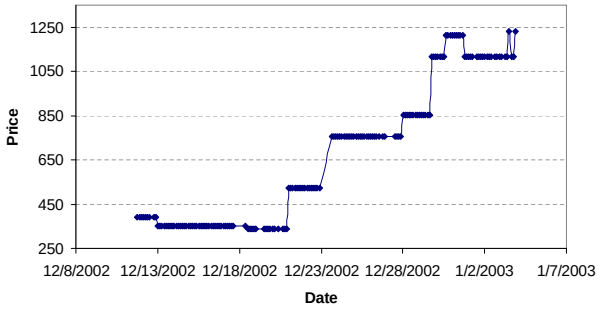

Figure 4: Price change over time for Alaska Airlines roundtrip flight#6:3, SEA-IAD departing on Jan 4. This figure shows an example of four price tiers.

need to submit the change to the Airline Tariff Publishing Company(ATPCO),[3] the organization formed by leading airlines around the world that collects and distributes airline pricing data. The whole process of detecting competitors' fare changes, deciding whether or not to match competitors' prices, and submitting the price update at ATPCO can take up to one day [19].

Price changes appear to be fairly orderly on some flights (e.g., Figure 3), and we see evidence of the well-known 7 and 14 day "advance purchase" fares. However, we also see plenty of surprising price changes. For example, flights that depart around holidays appear to fluctuate more (e.g., Figure 2. Figure 2 and Figure 3 show how pricing strategies differ between two flights from American Airlines that have the same schedule but fly on different dates. Figure 2 shows a flight that departs around the new year, while Figure 3 shows the flight that departs one week after the first flight. Both flights have the tier structure that we described earlier in this section, but ticket prices in the first flight fluctuate more often.

In terms of pricing strategy, we can divide the airlines into two categories. The first category covers airlines that are big players in the industry, such as United Airlines, and American Airlines. The second category covers smaller airlines that concentrate on selling low-price tickets, such as Air Trans and Southwest. We have found that pricing policies tend to be similar for airlines that belong to the same category. Fares for airlines in the first category are expensive and fluctuate often, while fares for airlines in the second category are moderate and appear relatively stable. However, there are some policies that every airline seems to use. For example, airlines usually increase ticket prices two weeks before departure dates and ticket prices are at a maximum on departure dates.

3. RELATED WORK

Previous work in the AI community on the problem of predicting product prices over time has been limited to the Trading Agent Competition (TAC) [27]. In 2002, TAC focused on the travel domain. TAC relies on a simulator of airline, hotel, and ticket prices and the competitors build agents to bid on these. The problem is different from ours since the competition works as an auction (similar to Price-

[3]see `http://www.atpco.net`.

line.com). Whereas we gathered actual flight price data from the web, TAC simulates flight prices using a stochastic process that follows a random walk with an increasingly upward bias. Also, the TAC auction of airline tickets assumes that the supply of airline tickets is unlimited. Several TAC competitors have explored a range of methods for price prediction including historical averaging, neural nets, and boosting. It is difficult to know how these methods would perform if reconfigured for our price mining task.

There has been some recent interest in temporal data mining (see [23] for a survey). However, the problems studied under this heading are often quite different from our own (e.g., [1]). There has also been algorithmic work on time series methods within the data mining community (e.g., [4]). We discuss time series methods below.

Problems that are closely related to price prediction over time have been studied in statistics under the heading of "time series analysis" [7, 13, 9] and in computational finance [20, 22, 21] under the heading of "optimal stopping problems". However, these techniques have not been used to predict price changes for consumer goods based on data available over the web. Moreover, we combine these techniques with rule learning techniques to improve their performance.

Computational finance is concerned with predicting prices and making buying decisions in markets for stock, options, and commodities. Prices in such markets are not determined by a hidden algorithm, as in the product pricing case, but rather by supply and demand as determined by the actions of a large number of buyers and sellers. Thus, for example, stock prices tend to move in small incremental steps rather than in the large, tiered jumps observed in the airline data.

Nevertheless, there are well known problems in options trading that are related to ours. First, there is the early exercise of American Calls on stocks that pay dividends. The second problem is the exercise of American Puts on stocks that don't pay dividends. These problems are described in sections 11.12 and 7.6 respectively of [14]. In both cases, there may be a time before the expiration of an option at which its exercise is optimal. Reinforcement learning methods have been applied to both problems, and that is one reason we consider reinforcement learning for our problem.

Time series analysis is a large body of statistical techniques that apply to a sequence of values of a variable that varies over time due to some underlying process or structure [7, 13, 9]. The observations of product prices over time are naturally viewed as time series data. Standard data mining techniques are "trained" on a set of data to produce a predictive model based on that data, which is then tested on a separate set of test data. In contrast, time series techniques would attempt to predict the value of a variable based on its own history. For example, our moving average model attempts to predict the future changes in the price of a ticket on a flight from that flight's *own* price history.

There is also significant interest in bidding and pricing strategies for online auctions. For example, in [24] Harshit et al. use cluster analysis techniques to categorize the bidding strategies being used by the bidders. And in [17], Lucking-Reiley et al. explore the various factors that determine the final price paid in an online auction, such as the length of the auction, whether there is a reserve price, and the reputation of the seller. However, these techniques are not readily applicable to our price mining problem.

Comparison shopping "bots" gather price data available on the web for a wide range of products.[4] These are descendants of the Shopbot [11] which automatically learned to extract product and price information from online merchants' web sites. None of these services attempts to analyze and predict the behavior of product prices over time. Thus, the data mining methods in this paper complement the body of work on shopbots.

4. DATA MINING METHODS

In this section we explain how we generated training data, and then describe the various data mining methods we investigated: Ripper [8], Q-learning [25], and time series [13, 9]. We then explain how our data mining algorithm, HAMLET, combines the results of these methods using a variant of stacked generalization [26, 28].

Our data consists of price observations recorded every 3 hours over a 41 day period. Our goal is to learn whether to buy a ticket or wait at a particular time point, for a particular flight, given the price history that we have recorded. All of our experiments enforce the following essential temporal constraint: all the information used to make a decision at particular time point was recorded *before* that time point. In this way, we ensure that we rely on the past to predict the future, but not vice versa.

4.1 Rule Learning

Our first step was to run the popular Ripper rule learning system [8] on our training data. Ripper is an efficient separate and conquer rule learner. We represented each price observation to Ripper as a vector of the following features:

- Flight number.
- Number of hours until departure (denoted as hours-before-takeoff).
- Current price.
- Airline.
- Route (LAX-BOS or SEA-IAD).

The class labels on each training instance were 'buy' or 'wait'.

We considered a host of additional features derived from the data, but they did not improve Ripper's performance. We did not represent key variables like the number of unsold seats on a flight, whether an airline is running a promotion, or seasonal variables because HAMLET did not have access to this information. However, see Section 6 for a discussion of how HAMLET might be able to obtain this information in the future.

Some sample rules generated by Ripper are shown in Figure 5.

In our domain, classification accuracy is not the best metric to optimize because the cost of misclassified examples is highly variable. For example, misclassifying a single example can cost from nothing to upwards of $2,000. MetaCost [10] is a well-known general method for training cost-sensitive classifiers. In our domain, MetaCost will make a learned classifier either more conservative or more aggressive about waiting for a better price, depending on the cost

[4]See, for example, `froogle.google.com` and `mysimon.com`.

IF hours-before-takeoff >= 252 AND price >= 2223
AND route = LAX-BOS THEN *wait*

IF airline = United AND price >= 360
AND hours-before-takeoff >= 438 THEN *wait*

Figure 5: Sample Ripper rules.

of misclassifying a 'buy' as a 'wait' compared with the cost of misclassifying a 'wait' as a 'buy'. We implemented MetaCost with mixed results.

We found that MetaCost improves Ripper's performance by 14 percent, but that MetaCost hurts HAMLET's overall performance by 29 percent. As a result, we did not use MetaCost in HAMLET.

4.2 Q-learning

As our next step we considered Q-learning, a species of reinforcement learning [25]. Reinforcement learning seems like a natural fit because after making each new price observation HAMLET has to decide whether to buy or to wait. Yet the reward (or penalty) associated with the decision is only determined later, when HAMLET determines whether it saved or lost money through its buying policy. Reinforcement learning is also a popular technique in computational finance [20, 22, 21].

The standard Q-learning formula is:

$$Q(a,s) = R(s,a) + \gamma max_{a'}(Q(a',s'))$$

Here, $R(s,a)$ is the immediate reward, γ is the discount factor for future rewards, and s' is the state resulting from taking action a in state s. We use the notion of state to model the state of the world after each price observation (represented by the price, flight number, departure date, and number of hours prior to takeoff). Thus, there are two possible actions in each state: b for 'buy' and w for 'wait'.

Of course, the particular reward function used is critical to the success (or failure) of Q-learning. In our study, the reward associated with b is the negative of the ticket price at that state, and the state resulting from b is a terminal state so there is no future reward. The immediate reward associated with w is zero as long as economy tickets on the flight do not sell out in the next time step. We set $\gamma = 1$, so we do not discount future rewards.

To discourage the algorithm from learning a model that waits until flights sell out, we introduce a "penalty" for such flights in the reward function. Specifically, in the case where the flight does sell out at the next time point, we make the immediate reward for waiting a negative constant whose absolute value is substantially greater than the price for any flight. We set the reward for reaching a sold-out state to be $-300,000$. This setting can best be explained below, after we introduce a notion of equivalence classes among states.

In short, we define the Q function by

$$\begin{aligned} Q(b,s) &= -price(s) \\ Q(w,s) &= \begin{cases} -300000 & \text{if flight sells out after } s. \\ max(Q(b,s'), Q(w,s')) & \text{otherwise.} \end{cases} \end{aligned}$$

To generalize from the training data we used a variant of the averaging step described in [18]. More specifically, we defined an equivalence class over states, which enabled the algorithm to train on a limited set of observations of the class and then use the learned model to generate predictions for other states in the class.

To define our equivalence class we need to introduce some notation. Airlines typically use the same flight number (e.g., UA 168) to refer to multiple flights with the same route that depart at the same time on different dates. Thus, United flight 168 departs once daily from LAX to Boston at 10:15pm. We refer to a particular flight by a combination of its flight number and date. For example, UA168-Jan7 refers to flight 168 which departs on January 7th, 2003. Since we observe the price of each flight eight times in every 24 hour period, there are many price observations for each flight. We distinguish among them by recording the time (number of hours) until the flight departs. Thus, UA168-Jan7-120 is the price observation for flight UA168, departing on January 7, which was recorded on January 2nd (120 hours before the flight departs on the 7th). Our equivalence class is the set of states with the same flight number and the same hours before takeoff, but different departure dates. Thus, the states denoted UA168-Jan7-120 and UA168-Jan10-120 are in the same equivalence class, but the state UA168-Jan7-117 is not. We denote that s and s^* are in the same equivalence class by $s \sim s^*$.

Thus, our revised Q-learning formula is:

$$Q(a,s) = Avg_{s^* \sim s}(R(s^*,a) + max_{a'}(Q(a',s')))$$

The reason for choosing -300,000 is now more apparent: the large penalty can tilt the average toward a low value, even when many Q values are being averaged together. Suppose, for example, that there are ten training examples in the same equivalence class, and each has a current price of $2,500. Suppose now that in nine of the ten examples the price drops to $2,000 at some point in the future, but the flight in the tenth example sells out in the next state. The Q value for waiting in any state in this equivalence class will be $(-300,000 - 2,000 * 9)/10 = -31,800$, or still much less then the Q value for any equivalence class where no flight sells out in the next state. Thus the choice of reward for a flight that sells out will determine how willing the Q-Learning algorithm will be to risk waiting when there's a chance a flight may sell out. Using a hill climbing search in the space of penalties, we found -300,000 to be locally optimal.

Q-learning can be very slow, but we were able to exploit the structure of the problem and the close relationship between dynamic programming and reinforcement learning (see [25]) to complete the learning in one pass over the training set. Specifically, the reinforcement learning problem we face has a particularly nice structure, in which the value of $Q(b,s)$ depends only on the price in state s, and the value of $Q(w,s)$ depends only on the Q values of exactly one other state: the state containing the same flight number and departure date but with three hours less time left until departure. Applying dynamic programming is thus straightforward, and the initial training step requires only a single pass over the data. In order to compute averages over states in the same equivalence class, we keep a running total and a count of the Q values in each equivalence class. Thus, the reinforcement learning algorithm just makes a single pass over the training data, which bodes well for scaling the algorithm to much larger data sets.

The output of Q-learning is the learned policy, which determines whether to buy or wait in unseen states by mapping them to the appropriate equivalence class and choosing the

action with the lowest learned cost.

4.3 Time Series

Time series analysis is a large and diverse subfield of statistics whose goal is to detect and predict trends. In this paper, we investigated a first order moving average model. At time step t, the model predicts the price one step into the future, p_{t+1}, based on a weighted average of prices already seen. Thus, whereas Q-learning and Ripper attempt to generalize from the behavior of a set of flights in the training data to the behavior of future flights, the moving average model attempts to predict the price behavior of a flight in the test data based on its own history.

At time t, we predict the next price using a fixed window of price observations, $p_{t-k+1}, \ldots, p_t$. (In HAMLET, we found that setting k to one week's worth of price observations was locally optimal.) We take a weighted average of these prices, weighting the more recent prices more and more heavily. Formally, we predict that p_{t+1} will be

$$\frac{\sum_{i=1}^{k} \alpha(i) p_{t-k+i}}{\sum_{i=1}^{k} \alpha(i)}$$

where $\alpha(i)$ is some increasing function of i. We experimented with different α functions and chose a simple linearly increasing function.

Given the time series prediction, HAMLET relies on the following simple decision rule: if the model predicts that $p_{t+1} > p_t$, then *buy*, otherwise *wait*. Thus, our time series model makes its decisions based on a one-step prediction of the ticket price change. The decision rule ignores the magnitude of the difference between p_{t+1} and p_t, which is overly simplistic, and indeed the time series prediction does not do very well on its own (see Table 3). However, HAMLET uses the time series predictions extensively in its rules. In effect, the time series prediction provides information about how the current price compares to a local average, and that turns out to be valuable information for HAMLET.

4.4 Stacked Generalization

Ensemble-based learning techniques such as bagging [5], boosting [12], and stacking [26, 28], which combine the results of multiple generalizers, have been shown to improve generalizer accuracy on many data sets. In our study, we investigated multiple data mining methods with very different characteristics (Ripper, Q-learning, and time series) so it makes sense to combine their outputs.

We preferred stacking to voting algorithms such as weighted majority [16] or bagging [5] because we believed that there were identifiable *conditions* under which one method's model would be more successful than another. See, for example, the sample rule in Figure 6.

Standard stacking methods separate the original vector representation of training examples (*level-0* data in Wolpert's terminology), and use the class labels from each level-0 generalizer, along with the example's true classification as input to a meta-level (or *level-1*) generalizer. To avoid over-fitting, "care is taken to ensure that the models are formed from a batch of training data that does not include the instance in question" [26].

In our implementation of stacking, we collapsed level-0 and level-1 features. Specifically, we used the feature representation described in Section 4.1 but added three additional features corresponding to the class labels (buy or wait) computed for each training example by our level-0 generalizers. To add our three level-1 features to the data, we applied the model produced by each base-level generalizer (Ripper, Q-learning, and time series) to each instance in the training data and labeled it with 'buy' or 'wait'. Thus, we added features of the form TS = buy (time series says to buy) and QL = wait (Q-learning says to wait).

Let TS be the output of the Time Series algorithm, and let QL be the output of Q-Learning.

IF hours-before-takeoff >= 480 AND $airline = United$ AND $price >= 360$ AND $TS = buy$ AND $QL = wait$ THEN *wait*

Figure 6: A sample rule generated by Hamlet.

We then used Ripper as our level-1 generalizer, running it over this augmented training data. We omitted leave-one-out cross validation because of the temporal nature of our data. Although a form of cross validation is possible on temporal data, it was not necessary because each of our base learners did not appear to overfit the training data.

Our stacked generalizer was our most successful data mining method as shown in Table 3 and we refer to it as HAMLET.

4.5 Hand-Crafted Rule

After we studied the data in depth and consulted with travel agents, we were able to come up with a fairly simple policy "by hand". We describe it below, and include it in our results as a baseline for comparison with the more complex models produced by our data mining algorithms.

The intuition underlying the hand-crafted rules is as follows. First, to avoid sell outs we do not want to wait too long. By inspection of the data, we decided to *buy* if the price has not dropped within 7 days of the departure date. We can compute an expectation for the lowest price of the flight in the future based on similar flights in the training data.[5] If the current price is higher than the expected minimum then it is best to wait. Otherwise, we buy.

More formally, let $MinPrice(s,t)$ of a flight in the training set denote the minimum price of that flight over the interval starting from s days before departure up until time t (or until the flight sells out). Let $ExpPrice(s,t)$ for a particular flight number denote the average over all $MinPrice(s,t)$ for flights in the training set with that flight number. Suppose a passenger asks at time t_0 to buy a ticket that leaves in s_0 days, and whose current price is $CurPrice$. The hand-crafted rule is shown in Figure 7.

IF $ExpPrice(s_0, t_0) < CurPrice$ AND $s_0 > 7$ days THEN *wait* ELSE *buy*

Figure 7: Hand-crafted rule for deciding whether to wait or buy.

We also considered simpler decision rules of the form "if the current time is less than K days before the flight's departure then buy." In our simulation (described below) we

[5]For "similar" flights we used flights with the same airline and flight number.

tested such rules for K ranging from 1 to 22, but none of these rules resulted in savings and some resulted in substantial losses.

5. EXPERIMENTAL RESULTS

In this section we describe the simulation we used to assess the savings due to each of the data mining methods described earlier. We then compare the methods in Table 3, perform a sensitivity analysis of the comparison along several dimensions, and consider the implications of our pilot study.

5.1 Ticket Purchasing Simulation

The most natural way to assess the quality of the predictive models generated by the data mining methods described in Section 4 is to quantify the savings that each model would generate for a population of passengers. For us, a passenger is a person wanting to buy a ticket on a particular flight at a particular date and time. It is easy to imagine that an online travel agent such as Expedia or Travelocity could offer discounted fares to passengers on its web site, and use HAMLET to appropriately *time* ticket purchases behind the scenes. For example, if HAMLET anticipates that a fare will drop by $500, the agent could offer a $300 discount and keep $200 as compensation and to offset losses due prediction errors by HAMLET.

Since HAMLET is not yet ready for use by real passengers, we simulated passengers by generating a uniform distribution of passengers wanting to purchase tickets on various flights as a function of time. Specifically, the simulation generated one passenger for each fare observation in our set of test data. The total number of passengers was 4,488. Thus, each simulated passenger has a particular flight for which they need to buy a ticket and an earliest time point at which they could purchase that ticket (called the "earliest purchase point"). The earliest purchase points, for different simulated passengers, varied from 21 days before the flight to the day of the flight.

At each subsequent time point, HAMLET decides whether to buy a ticket immediately or to wait. This process continues until either the passenger buys a ticket or economy seats on the flight sell out, in which case HAMLET will buy a higher priced business-class ticket for the flight.[6] We defined upgrade costs as the difference between the cost of a business class ticket and the cost of an economy ticket at the earliest purchase point. In our simulation, HAMLET was forced to "upgrade" passengers to business class only 0.42% of the time, but the total cost of these upgrades was quite high ($38,743 in Table 3).[7]

We recorded for each simulated passenger, and for each predictive model considered, the price of the ticket purchased and the optimal price for that passenger given their earliest time point and the subsequent price behavior for that flight. The savings (or loss) that a predictive model yields for a simulated passenger is the difference between the price of a ticket at the earliest purchase point and the price of the ticket at the point when the predictive model recommends buying. Net savings is savings net of both losses and upgrade costs.

[6]It's possible, of course, for business class to sell out as well, in which case HAMLET would have to buy a first-class ticket or re-book the passenger on a different flight. However, business class did not sell out in our simulation.

[7]Since we did not collect upgrade costs for all flights, our upgrade costs are approximate but always positive and often as high as $1,000 or more.

5.2 Savings

Table 3 shows the savings, losses, upgrade costs, and net savings achieved in our simulation by each predictive model we generated. We also report on the frequency of upgrades as a percentage of the total passenger population, the net savings as a percent of the total ticket price, and the performance of each model as a percent of the maximal possible savings.

The models we used are the following:

- **Optimal:** This model represents the maximal possible savings, which are computed by a "clairvoyant" algorithm with perfect information about future prices, and which obtained the best possible purchase price for each passenger.
- **By hand:** This model was hand-crafted by one of the authors after consulting with travel agents and throughly analyzing our training data (see Figure 7).
- **Time series:** This model was generated by the moving average method described earlier.
- **Ripper:** This model was generated by Ripper.
- **Q-learning:** This model was generated by our Q-learning method.
- **Hamlet:** This model was generated by our stacking generalizer which combined the results of Ripper, Q-learning, and Time series.

Table 3 shows a comparison of the different methods. Note that the savings measure we focus on is savings net of losses and upgrade costs. We see that HAMLET outperformed each of the learning methods as well as the hand-crafted model to achieve a net savings of $198,074. Furthermore, despite the fact that HAMLET had access to a very limited price history and no information about the number of unsold seats on the flight, its net savings were a remarkable 61.8% of optimal. Finally, while an average net savings of 4.4% may not seem like much, passengers spend billions of dollars on air travel each year so 4.4% amounts to a substantial number of dollars.

We believe that our simulation understates the savings that HAMLET would achieve in practice. For close to 75% of the passengers in our test set, savings were not possible because prices never dropped from the earliest purchase point until the flight departed. We report the percent savings in ticket prices over the set of flights where savings was possible ("feasible flights") in Table 4. These savings figures are of interest because of the unrealistic distribution of passengers in our simulation. Because we only gathered data for 21 days before each flight in our test set, passengers "arrived" at most 21 days before a flight. Furthermore, due to the uniform distribution of passengers, 33% of the passengers arrived at most 7 days before the flight's departure, when savings are hard to come by. In fact, on our test data, HAMLET lost money for passengers who "arrived" in the last 7 days prior to the flight. We believe that in practice we would find additional opportunities to save money for the bulk of passengers who buy their tickets more than 7 days before the flight date.

Method	Savings	Losses	Upgrade Cost	% Upgrades	Net Savings	% Savings	% of Optimal
Optimal	$320,572	$0	$0	0%	$320,572	7.0%	100%
By hand	$228,318	$35,329	$22,472	0.36%	$170,517	3.8%	53.2%
Ripper	$211,031	$4,689	$33,340	0.45%	$173,002	3.8%	54.0%
Time Series	$269,879	$6,138	$693,105	33.0%	-$429,364	-9.5%	-134%
Q-learning	$228,663	$46,873	$29,444	0.49%	$152,364	3.4%	47.5%
Hamlet	**$244,868**	**$8,051**	**$38,743**	**0.42%**	**$198,074**	**4.4%**	**61.8%**

Table 3: Savings by Method.

Method	Net Savings
Optimal	30.6%
By hand	21.8%
Ripper	20.1%
Time Series	25.8%
Q-learning	21.8%
Hamlet	**23.8%**

Table 4: Comparison of Net Savings (as a percent of total ticket price) on Feasible Flights.

5.3 Sensitivity Analysis

To test the robustness of our results to changes in our simulation, we varied two key parameters. First, we changed the distribution of passengers requesting flight tickets. Second, we changed the model of a passenger from one where a passenger wants to purchase a ticket on a particular flight to one where a passenger wants to fly at any time during a three hour interval. The interval model is similar to the interface offered at many travel web sites where a potential buyer specifies if they want to fly in the morning, afternoon, or evening.

We used the following distributions to model the earliest purchase point (i.e., the first time point at which passengers "arrive" and need to decide whether to buy a ticket or to wait):

- **Uniform:** a uniform distribution of simulated passengers over the 21 days before the flight's departure date;
- **Linear Decrease:** a distribution in which the number of passengers arriving at the system decreased linearly as the amount of time left before departure decreased;
- **Quadratic Decrease:** a distribution like Linear Decrease, but with a quadratic relationship;
- **Square Root Decrease:** a distribution like Linear Decrease, but with a square root relationship;
- **Linear Increase:** a distribution like Linear Decrease, except that the number of passengers increase as the amount of time left before departure decreased;
- **Quadratic Increase:** a distribution like Linear Increase, but with a quadratic relationship;
- **Square Root Increase:** a distribution like Linear Increase, but with a square root relationship.

Table 6 reports the net savings, as a percentage of the total ticket price, under the different distributions. HAMLET saved more than 2.5% of the ticket price in all cases, and it saved more than any other method on all distributions except the Quadratic Decrease distribution, where it performed slightly worse than the hand-crafted decision rule. HAMLET's savings were above 38% of optimal in all cases.

Table 5 reports on the performance of the different methods under the modified model where a passenger requests a ticket on a non-stop flight that departs at any time during a particular three hour interval (e.g., morning). This different model does not change our results qualitatively. HAMLET still achieves a substantial percentage of the optimal savings (59.2%) and its percentage of upgrades drops to only 0.1%. Finally, HAMLET still substantially outperforms the other data mining methods.

Method	Net Savings	% of Optimal	% upgrades
Optimal	$323,802	100%	0%
By hand	$163,523	55.5%	0%
Ripper	$173,234	53.5%	0%
Time Series	-$262,749	-81.1%	6.3%
Q-Learning	$149,587	46.2%	0.2%
Hamlet	**$191,647**	**59.2%**	**0.1%**

Table 5: Performance of algorithms on multiple flights over three hour interval.

Overall, our analysis confirms that HAMLET's performance on the test data is robust to the parameters we varied.

6. FUTURE WORK

There are several promising directions for future work on price mining. We plan to perform a more comprehensive study on airline pricing with data collected over a longer period of time and over more routes. We plan to include multi-leg flights in this new data set. The pricing behavior of multi-leg flights is different than that of non-stop flights because each leg in the flight can cause a change in the price, and because pricing through airline hubs appears to behave differently as well.

We also plan to exploit other sources of information to further improve HAMLET's predictions. We do not currently have access to a key variable — the number of unsold seats on a flight. However, on-line travel agents and centralized reservation systems such as Sabre or Galileo do have this information. If we had access to the number of unsold seats on a flight, HAMLET could all but eliminate the need to upgrade passengers, which is a major cost.

To use the methods in this paper on the full set of domestic and international flights on any given day would require collecting vast amounts of data. One possible way to address this problem is to build agents on demand that collect the required data to make price predictions for on a particular future flight on a particular day. The agents would still need

Distribution	By hand	Q-Learn	Time Series	Ripper	**Hamlet**
Quadratic Decrease	4.07%	3.77%	-24.96%	2.46%	**3.96%**
Linear Decrease	4.70%	4.30%	-26.76%	4.13%	**5.17%**
Sqrt Decrease	4.47%	4.04%	-29.05%	4.23%	**5.03%**
Uniform	3.77%	3.37%	-32.55%	3.83%	**4.38%**
Sqrt Increase	3.66%	3.24%	-34.63%	4.05%	**4.39%**
Linear Increase	3.13%	2.72%	-36.55%	3.62%	**3.85%**
Quadratic Increase	2.10%	1.74%	-39.90%	2.48%	**2.60%**

Table 6: Sensitivity of Methods to Distribution of Passengers' Earliest Purchase Points. The numbers reported are the savings, as a percentage of total ticket price, achieved by each algorithm under each distribution. We see that Hamlet outperforms Q-learning, time series, and Ripper on all distributions.

to collect data for multiple flights, but the amount of data would be much smaller. This type of agent would fit well within the Electric Elves system [6, 2], which deploys a set of personalized agents to monitor various aspects of a trip. For example, Elves can notify you if your flight is delayed or canceled or let you know if there is an earlier connecting flight to your destination.

Beyond airline pricing, we believe that the techniques described in this paper will apply to other product categories. In the travel industry, hotels and car rental agencies employ many of the same pricing strategies as the airlines and it would be interesting to see how much HAMLET can save in these product categories. Similarly, online shopping sites such as Amazon and Wal-mart are beginning to explore more sophisticated pricing strategies and HAMLET will allow consumers to make more informed decisions. Finally, reverse auction sites, such as half.com, also provide an opportunity for HAMLET to learn about pricing over time and make recommendations about purchasing an item right away or waiting to buy it. In general, price mining over time provides a new dimension for comparison shopping engines to exploit.

We recognize that if a progeny of HAMLET would achieve wide spread use it could start to impact the airlines' (already slim) profit margins. Could the airlines introduce noise into their pricing patterns in an attempt to fool a price miner? While we have not studied this question in depth, the obvious problem is that changing fares on a flight in order to fool a price miner would impact *all* consumers considering buying tickets on that flight. If the price of a ticket moves up substantially, then consumers are likely to buy tickets on different flights resulting in a revenue loss for the airline. Similarly, if the price moves down substantially, consumers will be buying tickets at a discount resulting in a revenue loss again. Thus, to avoid these distortions, the airlines are forced to show the prices that they actually want to charge for tickets. Of course, there are more prosaic methods of trying to block a price miner such as placing prices inside GIF files or blocking the IP address of the price miner. However, an "industrial strength" price miner would not rely on "scraping" information from web sites, but would access a fare database directly.

7. CONCLUSION

This paper reported on a pilot study in "price mining" over the web. We gathered airfare data from the web and showed that it is feasible to predict price changes for flights based on historical fare data. Despite the complex algorithms used by the airlines, and the absence of information on key variables such as the number of seats available on a flight, our data mining algorithms performed surprisingly well. Most notably, our HAMLET data mining method achieved 61.8% of the possible savings by appropriately timing ticket purchases.

Our algorithms were drawn from statistics (time series methods), computational finance (reinforcement learning) and classical machine learning (Ripper rule learning). Each algorithm was tailored to the problem at hand (e.g., we devised an appropriate reward function for reinforcement learning), and the algorithms were combined using a variant of stacking to improve their predictive accuracy.

Additional experiments on larger airfare data sets and in other domains (e.g., hotels, reverse auctions) are essential, but this initial pilot study provides the first demonstration of the potential of price mining algorithms to save consumers substantial amounts of money using data available on the Internet. We believe that price mining of this sort is a fertile area for future research.

8. ACKNOWLEDGMENTS

We thank Haym Hirsh, John Moody, and Pedro Domingos for helpful suggestions. This paper is based upon work supported in part by the Air Force Office of Scientific Research under grant number F49620-01-1-0053 to USC. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

9. REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[2] J. L. Ambite, G. Barish, C. A. Knoblock, M. Muslea, J. Oh, and S. Minton. Getting from here to there: Interactive planning and agent execution for optimizing travel. In *Proceedings of the Fourteenth Conference on Innovative Applications of Artificial Intelligence (IAAI-2002)*, pages 862–869, AAAI Press, Menlo Park, CA, 2002.

[3] G. Barish and C. A. Knoblock. An efficient and expressive language for information gathering on the web. In *Proceedings of the AIPS-2002 Workshop on Is there life after operator sequencing? – Exploring real world planning*, pages 5–12, Tolouse, France, 2002.

[4] D. Berndt and J. Clifford. Finding patterns in time series: a dynamic programming approach. In U. Fayyad, G. Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[5] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[6] H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. V. Pynadath, T. A. Russ, and M. Tambe. Electric elves: Applying agent technology to support human organizations. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*, 2001.

[7] C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman and Hall, London, UK, 1989.

[8] W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proc. of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, July 9–12, 1995. Morgan Kaufmann.

[9] F. Diebold. *Elements of Forecasting*. South-Western College Publishing, 2nd edition, 2000.

[10] P. Domingos. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164, San Diego, CA, 1999. ACM Press.

[11] R. Doorenbos, O. Etzioni, and D. Weld. A scalable comparison-shopping agent for the World-Wide Web. In *Proc. First Intl. Conf. Autonomous Agents*, pages 39–48, 1997.

[12] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, Bari, Italy, 1996. Morgan Kaufmann.

[13] C. W. J. Granger. *Forecasting in Business and Economics*. Harcourt Brace, second edition, 1989.

[14] J. C. Hull. *Options, Futures, and Other Derivatives*. Prentice Hall College Div, 5th edition, 2002.

[15] C. A. Knoblock, K. Lerman, S. Minton, and I. Muslea. Accurately and reliably extracting data from the web: A machine learning approach. In P. S. Szczepaniak, J. Segovia, J. Kacprzyk, and L. A. Zadeh, editors, *Intelligent Exploration of the Web*, pages 275–287. Springer-Verlag, Berkeley, CA, 2003.

[16] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, February 1994.

[17] D. Lucking-Reiley, D. Bryan, N. Prasad, and D. Reeves. Pennies from ebay: The determinants of price in online auctions. Technical report, University of Arizona, 2000.

[18] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(1-3):159–195, 1996.

[19] S. McCartney. Airlines Rely on Technology To Manipuate Fare Structure. *Wall Street Journal*, November 3 1997.

[20] J. Moody and M. Saffell. Reinforcement learning for trading systems and portfolios. In *KDD*, pages 279–283, 1998.

[21] J. Moody and M. Saffell. Minimizing downside risk via stochastic dynamic programming. In Y. S. Abu-Mostafa, B. LeBaron, A. W. Lo, and A. S. Weigend, editors, *Computational Finance 1999*, Cambridge, MA, 2000. MIT Press.

[22] J. Moody and M. Saffell. Learning to trade via direct reinforcement. In *IEEE Transactions on Neural Networks, Vol. 12, No. 4*, 2001.

[23] J. F. Roddick and M. Spiliopoulou. A bibliography of temporal, spatial and spatio-temporal data mining research. *SIGKDD Explorations*, 1(1):34–38, 1999.

[24] H. S. Shah, N. R. Joshi, A. Sureka, and P. R. Wurman. Mining for bidding strategies on ebay. In *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2003.

[25] R. S. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[26] K. M. Ting and I. H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.

[27] M. P. Wellman, D. M. Reeves, K. M. Lochner, and Y. Vorobeychik. Price prediction in a trading agent competition. Technical report, University of Michigan, 2002.

[28] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.