On Optimizing Airline Ticket Purchase Timing

WILLIAM GROVES and MARIA GINI, University of Minnesota

Proper timing of the purchase of airline tickets is difficult even when historical ticket prices and some domain knowledge are available. To address this problem, we introduce an algorithm that optimizes purchase timing on behalf of customers and provides performance estimates of its computed action policy. Given a desired flight route and travel date, the algorithm uses machine-learning methods on recent ticket price quotes from many competing airlines to predict the future expected minimum price of all available flights. The main novelty of our algorithm lies in using a systematic feature-selection technique, which captures time dependencies in the data by using time-delayed features, and reduces the number of features by imposing a class hierarchy among the raw features and pruning the features based on in-situ performance. Our algorithm achieves much closer to the optimal purchase policy than other existing decision theoretic approaches for this domain, and meets or exceeds the performance of existing feature-selection methods from the literature. Applications of our feature-selection process to other domains are also discussed.

Categories and Subject Descriptors: I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents*

General Terms: Algorithms, Experimentation, Economics

Additional Key Words and Phrases: Airline ticket prices, price prediction, data mining, e-commerce, feature selection

ACM Reference Format:
William Groves and Maria Gini. 2015. On optimizing airline ticket purchase timing. ACM Trans. Intell. Syst. Technol. 7, 1, Article 3 (October 2015), 28 pages.
DOI: http://dx.doi.org/10.1145/2733384

1. INTRODUCTION

The conventional wisdom of airline ticket purchasing states that it is generally best to buy a ticket as early as possible to avoid the risk of price increase. As prices do generally increase dramatically before a flight's departure, this seems correct. However, the earliest purchase strategy only occasionally achieves the lowest-cost ticket. This article proposes a model for estimating the optimal time to buy airline tickets. The ultimate application is to autonomously make daily purchase decisions on behalf of airline ticket buyers to lower their costs.

This kind of optimal airline ticket-purchasing problem from the consumer's perspective is challenging primarily because prices can vary significantly on a daily basis, but consumers do not have sufficient information about pricing of specific routes and airlines. Prices do vary in this domain for several reasons. Sellers (airlines) make significant long-term investments in fixed infrastructure (airports, repair facilities),

Authors' addresses: W. Groves and M. Gini, Department of Computer Science and Engineering, 200 Union St SE, University of Minnesota, Minneapolis, MN, 55455; emails: {groves, gini}@cs.umn.edu.

DOI: http://dx.doi.org/10.1145/2733384

hardware (planes), and route contracts. The specific details of these long-term decisions are intended to roughly match the expected demand but often do not match it exactly. Dynamic setting of prices is the mechanism that airlines use to synchronize their individual supply and demand in order to attain the greatest revenue.

We assume that buyers are seeking the lowest price for their ticket, while sellers are seeking to keep overall revenue as high as possible to maximize profit. Simultaneously, each seller must consider the price movements of its competitors to ensure that its prices remain competitive to achieve sufficient (but not too high) demand. Both types of relationships (buyers to sellers and airline to airline) need to be considered to effectively optimize decision making from the buyer's point of view.

A central challenge in airline ticket purchasing is in overcoming the information asymmetry that exists between buyers and sellers. Airlines can leverage historical data to predict the future demand for each flight. Demand for a specific flight is likely to change over time and will also change due to the pricing strategy adopted by the airline. For buyers without access to historical price information, it is generally best to buy far in advance of a flight's departure. However, this is not always best since airlines will adjust prices downward if they want to increase sales.

Given a corpus of historical data, we propose a learning approach that computes policies that do much better than the earliest purchase strategy. The success of our method depends on several novel contributions:

(1) We leverage a developer-provided hierarchical structure of the features in the domain and use automated methods to select which features to include or prune. This technique, which we call *Developer-Guided Feature Selection*[1], computes a feature set that is more informative than existing feature-selection methods. The knowledge of precedence relationships between features is provided by the practitioner when building a prediction model. For a motivating discussion on the benefits of Developer-Guided Feature Selection, see Groves [2013].
(2) We capture temporal trends in the model by allowing time-delayed observations to supplement or replace the most recently observed value for each of the selected features. We call the time-delayed observations *lagged features*.

We demonstrate experimentally that our model performs better than the Bing Travel "Fare Predictor," the best commercial system currently available for airline fare prediction. In addition to airline ticket price prediction, our approach is applicable to many real-world domains, with properties such as the need to handle advance reservations, a range of customer values for the same product, or stock-perishable inventory [Elmaghraby and Keskinocak 2003]. Industries with these properties include hotel booking, railroad transport car rental, and broadcasting industries.

This article extends our previous work [Groves and Gini 2013b]. We provide an analysis of how airlines use competitive pricing and show how the market competitiveness on each airline route explains the difference in prediction performance of our model across routes. We extend our prediction method to handle specific customers' preferences, such as purchasing only nonstop flights and from a specific airline. We present a sensitivity analysis of how the hierarchical structure provided by the practitioner affects prediction performance. Further, we analyze the optimal purchase timing accuracy of different models.

[1]In our previous work [Groves and Gini 2013b], this method is called User-Guided Feature Selection. The change more accurately reflects the source of knowledge used in the bias.

2. BACKGROUND AND RELATED WORK

Briefly, we review the background on airline pricing to provide context for our work. We also outline how our work differs from existing work and how our method for feature selection guided by the modeler compares with existing feature-selection methods.

2.1. Airline Ticket Pricing

2.1.1. Airline Yield Management and Dynamic Pricing. Airlines determine the prices to offer for each flight through a process called yield management, which is designed to maximize revenue given constraints such as capacity and future demand estimates (for an overview, see Belobaba [1987] and Smith et al. [1992]). Mismatches between airplane size and passenger demand are equalized through pricing, which can adjust demand. Choosing optimal pricing on an entire airline network is complex because there are instances (in hub-and-spoke networks) when sacrificing revenue on a particular flight can increase overall revenue of the entire network.

The current state of yield management in the airline industry is a direct result of decisions made about regulation in the industry [Smith et al. 1992]. Due to regulatory changes in 1979, airlines became free to adjust the price for each seat without restriction. This allowed airlines to divide the seats for each flight into different "fare classes" and charge different prices for effectively the same service.

In traditional yield management, the lowest air ticket prices quoted to customers are based on the available seats in each fare class for a particular flight (or origin–destination pair in the case of multistop itineraries). An airline can adjust the rate-of-fill for a particular flight by moving seats between fare classes (e.g., by moving high-cost seats into lower-cost fare classes). These decisions are traditionally made by humans who take into account previous demand, current sales, and competition.

2.1.2. Low-Cost Airlines. The airline market has changed with the introduction of low-cost airlines (LCAs). A study of the pricing strategies developed by LCAs in the European air-travel market [Piga and Filippi 2002] reports that tickets purchased between 30 and 8 days prior to departure are more expensive than tickets bought in other periods. Tickets bought within a few days from departure can be significantly cheaper but are not always available due to demand. LCAs do not compete against conventional airlines on price alone. They also use horizontal product differentiation to minimize the necessity to compete on price. Specifically, LCAs use secondary airports (not significantly served by conventional airlines) and fly on schedules that are maximally distant from existing players.

A later investigation [Bachis and Piga 2011] shows that airlines that have a significant share of the traffic at an individual airport tend to have higher prices than other carriers at the same airport. Also, an airline having a large portion of traffic between two airports (one direct route) tends to have greater market power than an airline having a large portion of traffic between two airports without a direct route. There is greater substitutability on routes with one or more stops, thus market power is lower. We show how market competition on individual routes affects the performance of our prediction models (Section 5.7).

2.1.3. Strategic Sales and Game Theory in Pricing. There are several efforts in the game-theory community to model aspects of the airline ticket domain, usually for the purpose of understanding competitive market dynamics of the oligopoly of sellers. In Subramanian et al. [1999], a dynamic programming model is presented for determining optimal fare-class allocation (of four fare classes) on a single departure date and flight number. Valuable insights provided by this study are that booking limits do not need to change monotonically over time (may increase or decrease) and it may be better for

an airline to sell a lower fare class instead of a higher fare class, for instance, due to differences in cancellation characteristics. We show an example later in Figure 4.

A game-theory model of dynamic pricing for an oligopoly of sellers facing strategic customers, that is, buyers who will delay purchase if there is a high likelihood of lower prices later, is presented in Levin et al. [2009]. The work assumes perfect foresight: all parties (sellers and customers) can estimate perfectly future outcome probabilities and utilities. If even a portion of the population of buyers is strategic, revenue for the sellers is reduced and no strategic defenses can fully ameliorate this effect. The critical conclusion of this work is that the most effective method to inhibit the impact of strategic consumers is to reduce the amount of information available to buyers. This may explain in part why, in spite of the technical feasibility, few predictive tools are available to individual purchasers of airline tickets. It is important to note that we model competitive aspects only indirectly through price relationships and not by modeling capacity, demand, and inventory, as advocated by a game-theory approach.

Beyond information asymmetry, airlines also use pricing behavior to reduce the effectiveness of strategic behavior. The presence of strategic consumers has been cited as a factor in increased price volatility: high price volatility makes consumers less able to judge the "right" price, and price volatility makes consumers less sensitive to price increases. The psychological effects are clear, but Mantin and Gillen [2011] also indicate that some systematic behaviors, such as increased price volatility, can signal price changes. Those are the phenomena that we seek to leverage in our price predictions.

2.2. Optimal Purchase Timing

Our work has been inspired by Etzioni et al. [2003], in which several purchasing agents attempt to predict the optimal purchase time of an airline ticket for a specific flight. The agents determine the optimal purchase time within the last 21 days prior to departure for a specific flight in their collected dataset. The authors compute the purchasing policy (a sequence of wait/buy signals) for many unique simulated passengers with a specific target airline, target flight, and date of departure. The optimal policy (the sequence of buy/wait signals that leads to the lowest possible ticket price) is used as a benchmark for each simulated passenger and the cost of each alternative purchasing agent is computed. The aggregate result shows that, given these purchasing criteria, it is possible to save significantly. We understand that Bing Travel's Fare Predictor is a commercialized version of the models in Etzioni et al. [2003], so we use it as a benchmark in our experimental results (Section 5.2).

Our work is different because we model the aggregate cost of all the flights on any airline meeting the date of travel and origin–destination pair requirements. We also allow for some preference criteria, such as number of stops and choice of airline. We model purchases up to 60 days before departure (instead of 21), and compare our results against realistic financial benchmarks (including buying as early as possible). In addition, our model provides a regression estimate for the expected best price between the current date and departure date.

2.3. Feature Selection

From a technical perspective, this prediction problem that we address can be phrased as a machine-learning problem involving both many features (possible variables that are relevant to prediction) and temporal trends. For an overview of automated feature-selection techniques, see Guyon and Elisseeff [2003], Hall [1999], and Zhao and Liu [2011]. There are many data-driven feature-selection techniques applicable to this domain [Molina et al. 2002]. Correlation-based Feature Selection (CFS) [Hall 2000] performs a filter-based feature selection using normalized Pearson's correlation. The

Table I. Airline Price Quote Specifications for All Airlines for Specific 5-Day Round Trips

	Example 1	Example 2
Quote date	13 May 2011	13 May 2011
Origin city	SEA	NYC
Destination city	IAD	LAX
Departure date	20 May 2011	20 May 2011
Return date	25 May 2011	25 May 2011
No. of itineraries returned	1135	1304
No. of airlines quoting itineraries	9	13

Note: The exact dates and cities shown are for Illustration purposes only.

algorithm starts with an empty feature set and uses forward best-first search (BFS) to incrementally add features. Wrapper-based methods employing search, such as BFS coupled with a machine-learning algorithm, have also been employed [Kohavi and John 1997]. Both techniques are included in our results for benchmarking.

In the context of the feature-selection literature, our Developer-Guided Feature Selection is classified as a wrapper-based approach because it uses prediction performance of the underlying machine-learning model measured on many feature subsets to find a well-performing feature set. The feature selection bias induced by the constraints is a way of addressing the bias-variance trade-off when building prediction models from data [Hastie et al. 2001]. Beyond feature selection, using prior domain knowledge to guide model calibration is also an emerging approach used for improving learning of Bayesian network structures from data (e.g., Zhou et al. [2014]).

3. DATA SOURCES

The data for our analysis were collected as daily price quotes from a major travel search website between February 22 and June 10, 2011 (109 observation days) for 7 different origin–destination pairs[2] for a total of 23.5 million price quotes. The specific pairs were selected to include major cities in different parts of the United States and some international destinations. A web spider was used to query for each pair of origin–destination and departure date, thus the results are representative of what a customer could observe.

We split this set sequentially into 3 datasets with the following lengths: 48, 20, and 41 days. The three periods are utilized as the training set, calibration set, and test set, respectively. These values were chosen so that the calibration set had at least 3 complete departures of each type (Monday and Thursday), the test set had at least 6 departures of each type, and the remainder forms the training set. Each query returned, on average, 1,200 unique round-trip itineraries from all airlines; most queries returned results from more than 10 airlines. Example queries for individual routes and dates are in Table I. Queries were made at the same time each day for consistency.

Bing Travel, a popular travel search website, has a Fare Predictor tool that provides a daily buy/wait policy recommendation for many routes and departure dates. We obtained these recommendations daily from the site for the test set period. Those recommendations are directly compared with our results in Section 5.

3.1. Pricing Patterns in Historical Data

There are strong cyclic patterns in the time series of prices. For example, Figure 1 shows the mean lowest price quoted by all airlines for a specific origin–destination pair

[2]The 7 routes are the following origin–destination pairs: HOU-NYC, MSP-NYC, NYC-CDG, NYC-CHI, NYC-HKG, NYC-MSP, and SEA-IAD. The airport codes for the cities are: HOU = Houston, TX, USA; MSP = Minneapolis, MN, USA; NYC = New York, USA; CDG = Paris, France; CHI = Chicago, IL, USA; HKG = Hong Kong, China; SEA = Seattle, WA, USA; IAD = Washington, DC, USA.

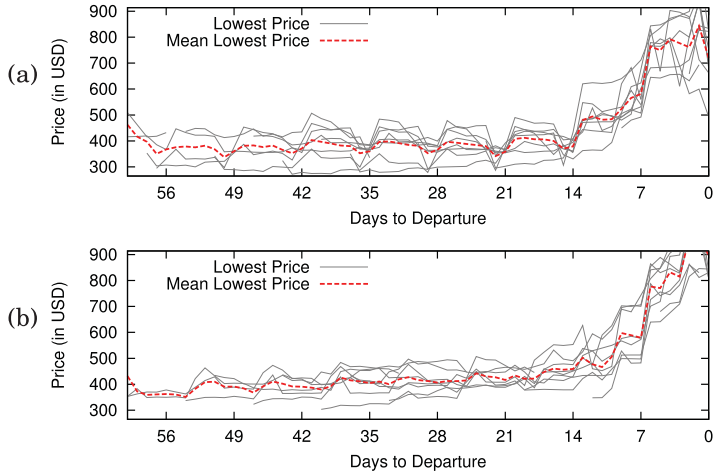

Fig. 1. Mean lowest price from all airlines for New York City (NYC) to Minneapolis (MSP) 5-day round-trip flights having (a) Thursday departure and Tuesday return or (b) Monday departure and Friday return itineraries. Each solid line series indicates the minimum price from all airlines on each query day for each departure date (8 departure dates in each graph). A dotted series indicates the mean.

for 2 months of 5-day round-trip itineraries departing on (a) Thursdays and (b) Mondays. The Thursday to Tuesday itinerary time series shows a regular drop in prices for Tuesday, Wednesday, and Thursday purchases (*days-to-departure* modulo $7 \in \{0, 1, 2\}$), while the (b) series shows significant increases for Thursday, Friday, and Saturday purchases. As expected, both exhibit price increases in the last few days before departure (*days-to-departure* ≤ 7), but series (b) exhibits this increase earlier in the time series. We posit that the majority of business flights would be Monday to Friday itineraries, thus demand for series (b) flights would be less sensitive to price increases than leisure flights. The weekly depression in costs in (a) may be due to market segmentation: customers buying mid-week are more price sensitive than weekend purchasers. Previous studies of airline pricing confirm a similar "weekend effect" in pricing [Mantin and Koo 2010; Puller and Taylor 2012].

The pricing behaviors exhibited for other origin–destination pairs also differ. A high traffic origin–destination pair such as the New York City to Los Angeles route (shown in Figure 2) exhibits much weaker cyclic patterns. Strategic pricing is likely to have a much greater observed effect for routes that have relatively few (2 or 3) competing airlines than for routes with a large number (>3) of competitors [Vowles 2000].

3.2. Observed Pricing Relationships

In this section, we characterize the pricing relationships between airlines that are observed in the pricing data that we collected. The prices quoted each day for a specific query (such as the two examples in Table 1) often vary significantly by airline, but the prices observed have structural relationships that can be leveraged for prediction. Figure 3 plots the minimum price time series for each airline from 4wk of data for a specific itinerary: depart Minneapolis (MSP) on May 5, 2011, and return from New York (NYC) on May 10, 2011. Pricing patterns for competing airlines have been covered empirically in the literature (e.g., Bachis and Piga [2011]); the figure illustrates these relationships.

Airlines can be clustered into categories according to their observed pricing strategies [Obeng and Sakano 2012]: low-cost airlines ("Low" category) and "legacy" airlines ("Mid" and "High" categories). Low category airlines use their primary advantage, the

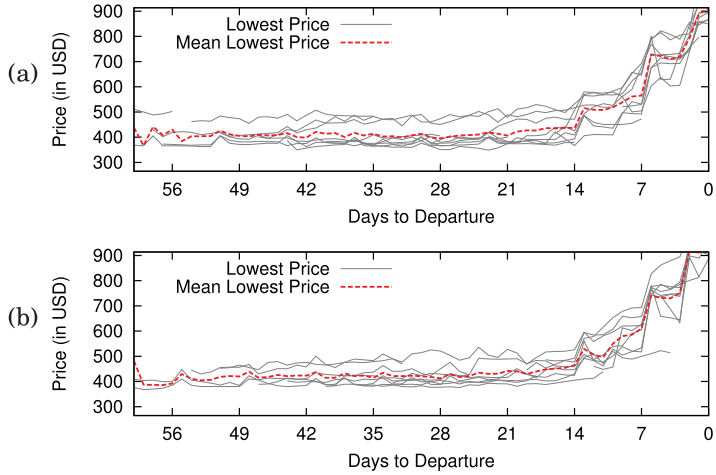

Fig. 2. Mean lowest price offered by all airlines for New York City (NYC) to Los Angeles (LAX) 5-day round-trip flights having (a) Thursday departure and Tuesday return, or (b) Monday departure and Friday return itineraries.

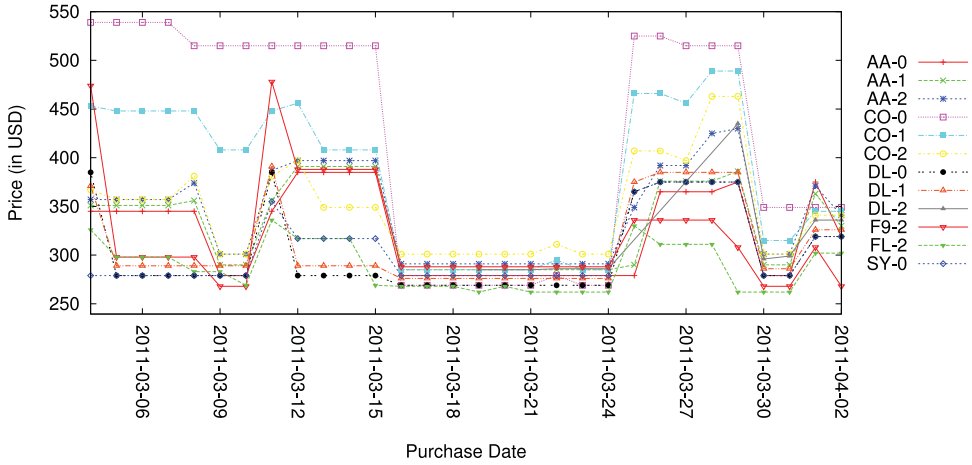

Fig. 3. A price time series for many airlines quoting prices for the NYC-MSP route for Thursday departure (May 05, 2011) with Tuesday return. Each series represents the minimum price of the day's quotes for a specific airline and number of stops per pair: for example, **DL-0** refers to all Delta Airlines nonstop flights and **AA-2** refers to all American Airlines, 2 intermediate stop flights.

ability to offer lower ticket prices due to lower internal costs, to compete aggressively against legacy airlines. Legacy airlines use other benefits to compete against Low category carriers, such as greater availability of departure times, a larger network of connecting flights, and loyalty rewards programs [Francis et al. 2006].

For a specific route's prices (shown in Figure 3), the competing airlines can be divided into one of three categories based on the pricing behavior. Low category airlines (referred to as low-cost airlines in the literature) tend to always compete on price and will consistently offer prices at or below all other types. Such airlines may even compete against other Low category carriers by lowering prices further in order to increase demand for the product. Mid-level airlines are legacy airlines that tend to price aggressively for the route but will rarely set prices below the best Low category price. High

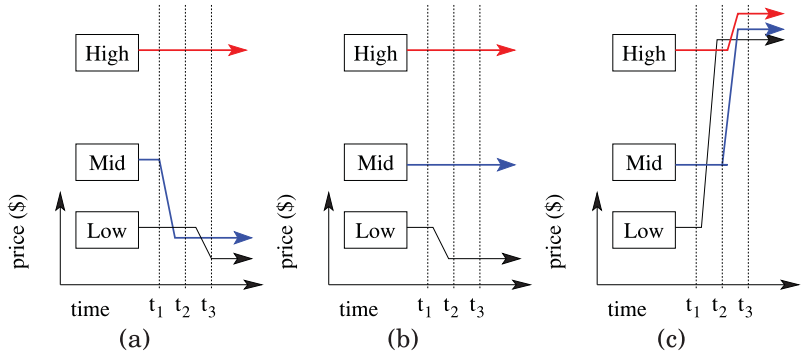

Fig. 4. Stylized diagrams of changes in price equilibria. Three price behavior types are shown: "Low" refers to low-cost airlines, "Mid" refers to medium-cost airlines, and "High" refers to high-cost airlines.

category airlines are legacy airlines that do not compete aggressively based on price but will still quote (usually higher) prices for the route. Customers may still buy from Mid or High categories because of the other benefits of the specific airline or itinerary.

In this route, the airlines can be categorized based on price behavior as:

—Low category: Frontier (F9), AirTran (FL), Sun Country (SY)
—Mid category: Delta (DL)
—High category: American (AA), Continental (CO)

Some airlines will quote itineraries with a different number of intermediate stops. For example, Delta Airlines (DL) quotes itineraries with no intermediate stops (nonstop, as DL-0), 1 intermediate stop (DL-1), or 2 intermediate stops (DL-2). For the purposes of the analysis, we treat each itinerary type separately, as shown in Figure 3.

The price time series in Figure 3 shows repeated patterns among the airlines for this departure and route. The Low category airlines are consistently at the bottom of the price range and there is a cluster of Mid-level airlines that have similar prices during periods of price stability. To characterize some of the price shifts, Figure 4 provides some stylized examples. In Figure 4(a), the Mid-level cluster lowers its price below the prevailing Low category price; on the next day, the Low category adjusts its prices to match or beat the best price. In Figure 4(b), a Low category airline lowers its price, but no other airline follows. In Figure 4(c), the Low category raises its price and the Mid and High category airlines adjust their prices upward as well.

These categorizations can be made from a statistical analysis of the data. These relationships between price behavior of airlines can be leveraged using a regression model because, within a route, the relationships are persistent. A machine-learning model, such as those described later in Section 4.3, can leverage these relationships to make predictions about future prices.

4. PROPOSED MODEL

When constructing prediction models for real-world domains, practical complexities must be addressed to achieve good prediction results. Typically, there are too many sources of data (features). Limiting the set of features in the prediction model is essential for good performance, but prediction accuracy can be lost if relevant inputs are pruned. This is even more critical when the number of observations available is limited, as often occurs in real-world domains.

To meet this challenge, we construct a prediction model (Figure 5) that involves the following distinct steps, which we then describe in detail:

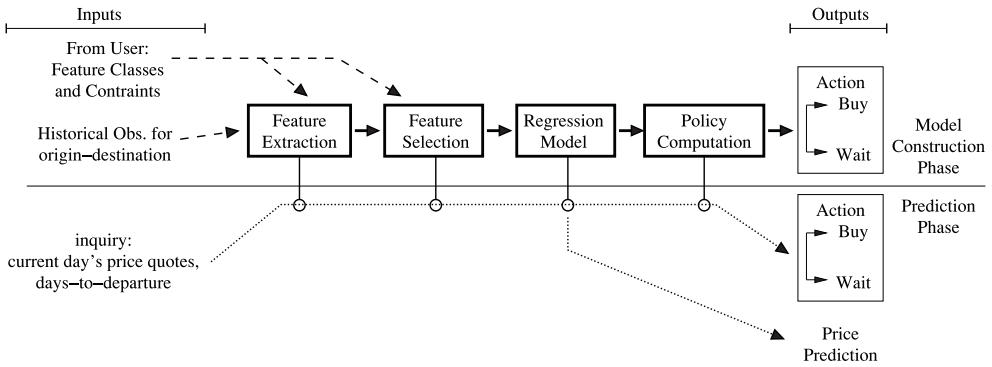

Fig. 5. Airline prediction model components, input data, and output.

(1) Feature Extraction and Feature Class Constraints – The raw data observed in the market are aggregated into a fixed-length feature set.
(2) Feature Selection and Lagged Features – A lag scheme is computed using a hierarchy of the features that incorporates some domain knowledge.
(3) Regression Model Construction – Using the augmented feature set generated from the lag scheme, a regression model is generated, for instance, using partial least squares (PLS) regression.
(4) Policy Computation – A search of decision threshold parameters is done to minimize cost on the calibration set.
(5) Optimal Model Selection – For each candidate model computed using the previous steps, the one that performs best on the calibration set is chosen. The final performance is estimated on the test set.

Figure 5 shows the distinct components of the model. Information used in the model construction phase includes historical data from the origin–destination pair and domain knowledge. The prediction phase for computing a buy/wait action at each decision day in the test set generates the action sequences scored in our experimental results.

4.1. Feature Extraction and Feature Class Constraints

The large number of itineraries (thousands) in each daily query made some data aggregation necessary. The features extracted are aggregated variables computed from the large list of prices from individual query days. For each query day, there are often many airlines quoting flights for that specific origin–destination and date combination. Airlines can vary their prices due to strategic decisions or to changes in available capacity.

To ensure a consistent, informative feature set, we build separate features for airlines quoting prices for 40% or more of the quote days in the dataset. For each airline exceeding the 40% criteria, its quotes are subdivided based on the number of intermediate stops: 0-stops ("nonstop"), 1-stop, or 2-stops. For each subdivision, three aggregate features are computed: the minimum price, mean price, and the number of quotes (corresponds to the EACH-STOPS feature class in Table II). These three aggregates are computed for the set of all the quotes from each airline (corresponds to the EACH-AGGREGATE feature class). Thus, for each airline, 12 features are computed on each quote day. For airlines not exceeding the 40% criteria, their itineraries are combined into a separate "OTHER" category placeholder. Finally, these same 12 aggregates are generated for all quotes and are placed in the "ALL" airlines category (corresponds to the ALL-STOPS and ALL-AGGREGATE feature classes). Boolean variables are added to

Table II. Raw Features by Feature Class for Each Quote Day for a Specific Departure Day and Route

Feature Class	Variable Count	Variable List
DETERMINISTIC	8 vars	Days-to-departure, Quote DoW is Mon, Quote DoW is Tue, . . . , Quote DoW is Sun
ALL-AGGREGATE	3 vars	ALL-min-A, ALL-mean-A, ALL-count-A
ALL-STOPS	9 vars	ALL-min-0, ALL-mean-0, ALL-count-0, ALL-min-1, ALL-mean-1, ALL-count-1, ALL-min-2, ALL-mean-2, ALL-count-2
EACH-AGGREGATE	18 vars	DL-min-A, DL-mean-A, DL-count-A, . . . , OTHER-min-A, OTHER-mean-A, OTHER-count-A
EACH-STOPS	54 vars	DL-min-0, DL-mean-0, DL-count-0, DL-min-1, DL-mean-1, DL-count-1, DL-min-2, DL-mean-2, . . .

Note: The number of variables in some classes (EACH-AGGREGATE, EACH-STOPS) will vary based on the number of airlines quoting the route. The counts given are specific to the NYC-MSP route (92 total raw variables). Variables are named as "<airline>-<statistic>-<#ofStops>": for example, ALL-min-A = minimum price quoted by any airline, ALL-min-0 = minimum price quoted by any airline for nonstop flights only, DL-min-A = minimum price quoted by a specific airline (DL = Delta Airlines). Variables named like "Quote DoW is Mon" are Boolean variables indicating the weekday the quote is retrieved.

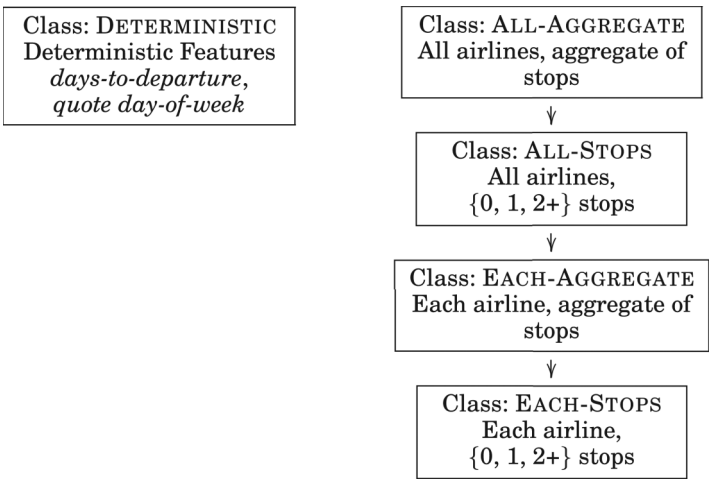

Fig. 6. Lag scheme class hierarchy for price prediction. Arrow denotes a *subset* relationship (e.g., class ALL-AGGREGATE should have an equal or greater set size than class ALL-STOPS).

indicate the query's weekday (for instance, "Quote DoW is Mon" is true if the quote is retrieved on Monday). A *days-to-departure* (number of days between the query and departure dates) value is computed based on the departure date.

Concurrent with the feature extraction process (when converting the thousands of itineraries for each query into feature values) is the computation of the target variable's value (e.g., minimum of ALL-min-A or minimum of DL-min-0). For example, in the computation of the DL-min-0 target variable on each query day, the minimum price of all nonstop tickets on all queries between the query date and the departure date is the value assigned for that observation day. This value represents the minimum possible cost ticket that could be purchased to satisfy the travel need given the preferences (DL = Delta Airlines, 0 = nonstop-only flights) and departure date.

A list of the features for a specific departure day and origin–destination for each query day is shown in Table II. Each of the 92 features is in a feature class based on its specificity using the feature class hierarchy in Figure 6. The number of variables in some classes (EACH-AGGREGATE, EACH-STOPS) will vary based on the number of

airlines quoting the route. Variables are named with the compact scheme "<airline>-<statistic>-<#ofStops>": for example, ALL-min-A = minimum price quoted by any airline, ALL-min-0 = minimum price quoted by any airline for nonstop flights only, DL-min-A = minimum price quoted by a specific airline (DL = Delta Airlines).

The feature classes are groupings of raw features in the domain. For this domain, there is a deterministic set of features ("DETERMINISTIC") that includes information about the number of days from the quote day until departure, and the 7 binary variables indicating the weekday of the quote (i.e., Monday, Tuesday, ...). At most, one instance of the DETERMINISTIC feature class is included because it contains deterministic information. Knowing the value for one day allows for all time lagged days' values to be determined exactly, thus there is no informational benefit to include more than one instance.

The ALL-AGGREGATE feature class contains raw features applicable to all airlines: the minimum price, mean price, and number of price quotes from any airline on the quote day for the origin–destination. The ALL-STOPS feature class is similar to ALL-AGGREGATE but has separate statistics based on the number of intermediate stops. This class is "more specific" than ALL-AGGREGATE; the data are more fine grained, thus we say that ALL-STOPS is more specific than ALL-AGGREGATE. This is indicated with a directed edge between the two in Figure 6. ALL-STOPS also contains more variables than ALL-AGGREGATE (9 vs. 3). Using similar arguments, the two more-specific classes EACH-AGGREGATE and EACH-STOPS are constructed. The size of EACH-AGGREGATE and EACH-STOPS (in terms of the number of significant competitors) will vary based on the origin–destination to be predicted. For the NYC-MSP route, there are five major competitors, thus there are 18 features in EACH-AGGREGATE[3]. For this domain, we constrain the feature classes to always include the more general classes before more-specific classes using the precedence hierarchy shown in Figure 6. Domain knowledge is used to build this structure. These precedence relationships prevent highly specific and possibly redundant information from being included in the augmented feature set before more general variables are included.

After feature extraction and identification of feature classes (shown in Table II), known hierarchical relationships between the feature classes will make up the constraint set. For the airline domain, the feature-class relationships in Figure 6 are common sense domain knowledge that can be elicited from even a novice practitioner with basic knowledge of the domain to bias the feature-selection search. For a discussion on alternative constraint sets, see Section 5.5.

4.2. Feature Selection and Lagged Features

Using only the most recent values (92 features for the NYC $\rightarrow$ MSP route) as the entire feature set may provide reasonable prediction results in some domains, but such a model cannot predict trends or temporal relationships present in the data. This simple scheme is shown in Table III(b) as *Most Recent Observations*.

The need to represent temporally offset relationships (such as weekly cycles or trends) motivates adding time-delayed observations to the feature set as well. This is accomplished by including past days' observed values in addition to the current day's value in the feature vector used for learning. We refer to this as the addition of *lagged features*. For instance, if the cost of a route on day $t - 7$ is representative of the price available on day $t + 1$, the 7-day delayed observation should have a high weight in the model. A regression model that includes all time-delayed instances up to a depth of n days of all features can produce good results, but the inclusion of too many variables into the model can result in poor performance. A diagram of this model is in Table III(a).

[3]Five major competitors plus the "other" category for small airlines and 3 features per airline: $(5+1)*3 = 18$.

Table III. Basic Lag Schemes used for Benchmarking the Feature Selection Methods

(a) Full Lag Scheme (lags are in days)

Class	Lagged Offsets							
	0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A	•	•	•	•	•	•	•	•
ALL-S	•	•	•	•	•	•	•	•
EACH-A	•	•	•	•	•	•	•	•
EACH-S	•	•	•	•	•	•	•	•

(b) Most Recent Observations

Class	Lagged Offsets							
	0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A	•							
ALL-S	•							
EACH-A	•							
EACH-S	•							

Note: The dots "•" indicate that the feature class at the corresponding time lag is included in the feature set provided to the learning model.

The performance of both *Full Lag Scheme* with ($n = 8$) and *Most Recent Observations* is shown in Table VI.

Lagged features, known as tapped delay lines in engineering, are well known in the time series literature [Sauer 1994] and have been used as input to machine-learning models [Wan 1994; Du and Swamy 2014]. The augmented feature set can be constructed from a generated lag scheme configuration. For each feature class and lagged offset pair containing a dot (•), all variables in the class are added to the augmented feature set for the specified lag offset.[4]

Our technique assumes that more recent observations are likely to have high informational value for price prediction, but time-delayed features may hold informational value as well (i.e., the environment is not completely stochastic). The time between a change in the market and its effect on the target variable may be longer than 1 day. Lagged variables can leverage those delayed relationships.

By examining all combinations of feature classes, we can automatically tune the feature vector to achieve better results. For each feature class, all contiguous subsets of lags and the empty set are examined as possible candidates. Another constraint is added due to relationships between classes: more-specific feature classes cannot have more time-lagged instances than more general classes (i.e., the more specific class's lags will be a subset of the more general class's lags).

Time-delayed observations from the target variable (such as the all airline minimum price in *Class* ALL-AGGREGATE) are likely to be most predictive because they are most general. Time-delayed observations from other more-specific feature classes *may also be* but are less likely to be predictive. The hierarchy and strict ordering of lagged data are based on this principle. By constraining the classes so that the less-informative classes contribute fewer features, we prevent the inclusion of irrelevant features.

[4]A brief example is provided: Given a lag scheme with dots only on the ALL-A feature class, if the ALL-A feature class has dots on lagged offsets 0, -1, and -2 only, then the augmented feature vector will contain 9 augmented variables that are (variable name, time offset) tuples. These 9 augmented variables will be: (ALL-min-A,0), (ALL-mean-A,0), (ALL-count-A,0), (ALL-min-A,-1), (ALL-mean-A,-1), (ALL-count-A,-1), (ALL-min-A,-2), (ALL-mean-A,-2), and (ALL-count-A,-2). The time offset refers to the number of days into the past from the quote day that the value should be extracted from the observations: (ALL-min-A,0) refers to today's value for ALL-min-A, and (ALL-min-A,-1) refers to yesterday's value for ALL-min-A.

Next, time-lagged data are used to form the augmented feature set. A search of all the possible lag configurations is done to find the best-performing configuration for the given departure date and origin–destination. Note that the optimal configuration may be different for each date and route.

The inclusion of a little domain knowledge using feature classes and the class hierarchy reduces significantly the number of possible feature set configurations that need to be searched. Without the hierarchy and constraints between features, there are $\approx 10^{82}$ configurations of the 92 original features.[5] Without the constraints between classes, the number of configurations of the 4 feature classes (having time delays of $\{\varnothing, 0, 1, 2, \ldots, 7\}$) would be very large at $\approx 1.9 * 10^6$, but many configurations are uninteresting variants.[6] The number of lag schemes formulated with the hierarchy and constraints in Figure 6 for a maximum time delay of 7 days is 8517.[7] Using both the feature classification and the constraint hierarchy allows for "interesting" lag schemes to be efficiently evaluated in a smaller number of evaluations.

The advantage of the automated lag scheme search is that it produces results similar to what a domain expert could do but using only minimal domain knowledge. Table IV shows optimal lag schemes for several routes. The results of the optimal lag scheme search can uncover some surprising relationships in the data. For example, it is interesting to note that nonstop targets in Table IV(b, d) benefit from a larger feature set (both in temporal depth and feature class breadth).

Before we can explain how the optimal lag scheme is selected, we need to describe the machine-learning methods that we use and how an action policy (buy or wait) is computed.

4.3. Machine-Learning Method

Our Developer-Guided Feature Selection is agnostic about the underlying machine-learning algorithm used for learning. In our experiments, we used many such algorithms. Results for a sample of well-studied machine-learning algorithms from the literature are shown in Table VI in Section 5.

Some of the learning methods that we use are classifiers and output a buy-or-wait action instead of a price. In this case, the learning method and decision threshold components are replaced with a classification algorithm or an external action strategy (such as Bing Travel's recommendations). This modification is indicated in the Learning Method Output column of Table VI by "Buy/Wait." The column contains "Regression" when instead a regression model is used.

As the feature-selection framework considers the underlying learning model as a black box, a wide range of time-series methods could be applied using this framework as well [Box et al. 2013]. Our experiments utilize Ripper and linear regression univariate time-series methods as points of comparison. For the majority of our experiments, we use a machine-learning regression or classification learner. The most compelling reason to use a multivariate regression/classification learner over a multivariate time-series learner is the following: the data for each route is actually made of a collection of shorter time-series. There is one time series for each departure date which has a length of at most 60 days. The frequent restarts required for standard time-series models make their application somewhat less natural. Specifically, it necessitates also choosing an initial state for the time-series model at each restart.

[5] 84 price features and 8 deterministic features (days-to-departure and weekday of quote) $= (2^8) \times (2^{84*8})$.
[6] For each of the 4 feature classes, there are 8 lags (plus $\varnothing$) yielding 36 contiguous subsets (plus one additional configuration for $\varnothing$). The count of possible combinations for 4 feature classes is $37^4 = 1874161$.
[7] The mathematical formula for finding the exact number of unique configurations among a chain of subset constraints (ALL-AGGREGATE to EACH-STOPS) is available in Appendix A.

Table IV. Optimal Lag Schemes of a Domestic Route and an International Route for a 5-Day Trip with Monday Departure

(a) New York → Minneapolis

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A	•	•	•	•	•	•	•	•
ALL-S		•	•	•	•	•	•	
EACH-A				•				
EACH-S								

(MC: $280, EP: $317, PAO: 4.6%)

(b) New York → Minneapolis (non-stop only)

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A	•	•	•	•	•			
ALL-S				•				
EACH-A				•				
EACH-S				•				

(MC: $365, EP: $414, PAO: 7.1%)

(c) New York → Hong Kong

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A	•	•	•	•	•	•	•	•
ALL-S								
EACH-A								
EACH-S								

(MC: $1190, EP: $1207, PAO: 3.8%)

(d) New York → Hong Kong (non-stop only)

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A	•	•	•	•	•	•	•	•
ALL-S	•	•	•	•	•	•	•	
EACH-A	•	•	•	•				
EACH-S	•	•	•	•				

(MC: $1404, EP: $1416, PAO: 6.1%)

Note: The dots "•" indicate that the feature class at the corresponding time lag is included in the best performing feature set. "MC" is the average model cost, "EP" the average earliest purchase cost, and "PAO" is the mean price of the model above the optimal policy price (in %).

For pure regression-based methods, we use support vector regression (nuSVR, [Schölkopf et al. 2000]), PLS regression, and ridge regression. The decision-tree classifier we use (REPTree, [Witten and Frank 2005]) can also predict values of continuous functions, thus both modes are shown in the experiments for comparison.

Another method we utilize in our experiments is a regression algorithm that creates a linear model: PLS regression. Mathematically, PLS regression deterministically computes a linear function that maps a vector of the input features $\vec{x}$ into the output variable y (the label) using a vector of weights $\vec{w}$. Several implementations of PLS exist [de Jong 1993; Martens and Næs 1992], each with its own performance characteristics. We use the orthogonalized PLS implementation in Wold et al. [1983].

We have chosen PLS over similar multivariate techniques including multiple linear regression, ridge regression [Hoerl and Kennard 2000], and principal component regression (PCR) [Jolliffe 1982] because of its advantages and better performance. First, PLS regression is able to handle very high-dimensional inputs because it implicitly

performs dimensionality reduction from the number of inputs to the number of PLS factors. Second, the model complexity can be adjusted by changing the number of PLS factors to use in computing the regression result. These factors are analogous to the principal component vectors used in principal component regression. The number of PLS factors determines the dimensionality of the intermediate variable space that the data is mapped to (we use a limit of 5 factors in our results). The computation time does not significantly increase for a larger number of factors but the choice can affect prediction performance: too many factors can cause overfitting, and too few factors can cause the model to be unable to represent relationships in the data. This value is adjusted in our experiments to determine the optimal model complexity in each prediction class. Finally, the algorithm is generally robust to highly collinear or irrelevant features.

4.4. Policy Computation and Evaluation

An obvious approach to choosing a good regression model (lag scheme and trained machine-learning model) is to use the model with the highest prediction accuracy, but this may not be the model that generates the lowest average cost policy. Instead, we rank the models by measuring the cost that results from following the computed policy recommendation. To use the regression output (an expected future price) to compute an action policy, we introduce the concept of a decision threshold function. Given $\hat{e}_t$, the model estimate future price at time t, the current observed price p_t and the current number of days-to-departure d_{dtd} (an integer), the current action policy $r_t \in \{\text{BUY}, \text{WAIT}\}$ is computed by Equation (1).

$$r_t = \begin{cases} \text{BUY} & : \hat{e}_t > p_t \times (c + (1/30) \times s \times d_{dtd}) \\ \text{WAIT} & : \text{otherwise} \end{cases} \tag{1}$$

The two parameters c and s are expressed as real numbers. Intuitively, c can be thought of as an adjustment in the likelihood of a BUY signal. Values of $c > 1.0$ correspond to a policy that is only likely to emit BUY when the current price is far below the expected future price. This situation indicates that the current price is a bargain for the customer. The parameter s corresponds to the percent change in the threshold per 30 days of advance purchase (0.02 corresponds to a 2% change in the threshold at $d_{dtd} = 30$). Values of $s > 0.0$ generate a policy more likely to WAIT when far from departure date. When a departure is far in the future and $s > 0.0$, the agent is more likely to wait until a highly favorable (low) price appears before deciding to purchase. Adjusting these two parameters can be thought of as determining the optimal level of risk depending on the current price and the degree of advance purchase. The range of c and s values searched in our experiments was $[0.7, 1.3]$ and $[-0.1, 0.1]$, respectively, in increments of 0.01.

We use this two-parameter approach to make decisions, because it is simple, works well, and provides an intuitive understanding of the policy computation. This is not to rule out more sophisticated approaches, such as reinforcement learning. We leave exploration of this aspect for future work.

4.5. Optimal Model Selection

The proposed search of lag schemes is exhaustive, but due to the feature class hierarchy, the number of configurations is relatively small and can be fully explored. The process is outlined in Algorithm 1.

A model is constructed for each potential lag scheme: first, a pricing model is generated for each lag scheme using the training set data, then the decision threshold parameters (c and s) are calibrated on the calibration set to discover the settings with the lowest average ticket price ($score_{\text{CA}}$). Performance ($score_{\text{TE}}$) is measured by applying the calibrated model to the test set.

ALGORITHM 1: Developer-Guided Feature Selection: Training, Calibration, and Testing Process

Data: historical price quotes for origin–destination (training set $hist_{\text{TR}}$, calibration set $hist_{\text{CA}}$, test set $hist_{\text{TE}}$), feature classes (fc) and feature class constraints (co)
Result: best observed lag scheme (ls_{best}), decision threshold parameters (s_{best}, c_{best}), calibration set mean cost ($score_{\text{CA}}$),test set mean cost ($score_{\text{TE}}$)

```
{ score_CA ← null,  ls_best ← null,  c_best ← null,  s_best ← null }
RM_TR, P_TR ← doFeatureExtraction(hist_TR, fc)
RM_CA, P_CA ← doFeatureExtraction(hist_CA, fc)
L ← generateLagSchemes(fc, co)
for ls in L do
    AM_TR ← buildAugmentedFeatureMatrix(RM_TR, ls)
    PriceModel ← trainRegressionModel(AM_TR, P_TR)
    P̂_CA ← predict(PriceModel, AM_CA)
    for c in {0.7, 0.71, ..., 1.3} do
        for s in {−0.1, 0.09, ..., 0.1} do
            score ← doPolicyComputationAndScore(AM_CA, P̂_CA, P_CA, c, s)
            if score_CA = null or score_CA > score then
                {score_CA ← score,  ls_best ← ls,  c_best ← c,  s_best ← s }
RM_TE, P_TE ← doFeatureExtraction(hist_TE, fc)
AM_TE ← buildAugmentedFeatureMatrix(RM_TE, ls_best)
P̂_TE ← predict(PriceModel, AM_TE)
score_TE ← doPolicyComputationAndScore(AM_TE, P̂_TE, P_TE, c_best, s_best)
```

5. EXPERIMENTAL RESULTS

The experiments were designed to estimate real-world costs using various prediction models to develop a purchase policy. A survey of the literature revealed that airlines assume a relatively fixed rate of purchases until a flight is full, and most tickets for a flight are sold within 60 days of departure [Belobaba 1987]. Using these facts, we measure performance as the cost of following the purchase recommendations for a specific departure once for purchases between 1 and 60 days before departure ($dtd \in \{1, 2, \ldots, 60\}$). This measure involves hypothetically purchasing an itinerary precisely 60 times for each purchase algorithm under test (but some purchases may be deferred for a few days based on the model output). Each of the 60 purchases is called a *purchase episode*. On each query day, the policy generator uses the currently observed prices (and possibly time-lagged observations) to compute a BUY or WAIT signal for the day. Using the sequence of BUY and WAIT signals, it is possible to determine the costs experienced by the algorithm for each simulated purchase. Mathematically, the cost for each simulated purchase experienced by a purchase "method" (e.g., Earliest Purchase, Optimal, and so on) is computed in Equation (2) where the r_is are obtained from one departure date.

$$\text{cost}_t = \begin{cases} p_t, & \text{where } r_t = \text{BUY} \\ \text{cost}_{t+1}, & \text{where } r_t = \text{WAIT} \end{cases}. \tag{2}$$

These costs are aggregated with data from all departure dates (DD is all departure dates in the test data set $hist_{\text{TE}}$) to compute the score for the method in Equation (3).

$$score_{\text{method}} = \text{mean}(\{\text{cost}_{dd,t} \mid dd \in \text{DD and } t \in T_{dd}\}) \tag{3}$$

T_{dd} is all time units applicable to departure date dd. By convention, the last day before departure is always labeled with a BUY signal, thus cost_t is always defined.

Table V. Actions Computed by Four Different Methods for up to 13 Days Before Departure for a Specific Request (NYC-MSP, Depart May 12, 2011, Return May 17, 2011)

Days to departure		13	12	11	10	9	8	7	6	5	4	3	2	1	0	**Score**
Current price (in $)		258	257	257	257	257	282	292	330	298	330	330	222	469	453	
Earliest purchase	action	•	•	•	•	•	•	•	•	•	•	•	•	•	•	**306.6**
	cost (in $)	258	257	257	257	257	282	292	330	298	330	330	222	469	453	
Our model	action	•	○	○	•	•	○	○	○	•	○	○	•	○	•	**289.3**
	cost (in $)	258	257	257	257	257	298	298	298	298	222	222	222	453	453	
Optimal	action	○	○	○	○	○	○	○	○	○	○	○	•	○	•	**255.0**
	cost (in $)	222	222	222	222	222	222	222	222	222	222	222	222	453	453	
Latest purchase	action	○	○	○	○	○	○	○	○	○	○	○	○	○	•	**453.0**
	cost (in $)	453	453	453	453	453	453	453	453	453	453	453	453	453	453	

Note: The ○ symbol indicates a "WAIT" signal for that day, and the • symbol indicates a "BUY" signal for that day. Themodels are compared and scored (last column) using the mean of all costs for simulated purchases. There is one simulated purchase for each day up to the departure date.

Table V shows examples of purchasing signals generated by four different policy generators (Earliest Purchase, Our Model, Optimal, and Latest Purchase) for a specific origin–destination pair and departure date. The performance of each policy generator is the mean of the cost values across all purchase episodes. By comparing purchasing signals, we can compute the percentage above the optimal cost (PAO) for each method (i.e., how far the method is above the optimal cost). The earliest purchase strategy represents the cost of buying at the first decision point in each purchase episode and will have a PAO of usually 5% to 30%. The optimal purchase strategy is the lowest possible cost for the purchase episode, which can be computed optimally in hindsight and represents a PAO of 0%. Well-performing policy generation algorithms should do at least as well as earliest purchase on almost all purchase episodes and should achieve costs as close to the optimal purchase as possible. Algorithms can be compared using the PAO value, well-performing methods will be between earliest purchase and optimal purchase value. Note that it is possible for an algorithm to have a PAO greater than the earliest purchase strategy due to having costs that are higher than an earliest purchase; latest purchase is an example that often achieves a much higher PAO score.

5.1. Model Comparison

The experiments are first categorized by the type of feature selection employed, then by the underlying machine-learning algorithm used. Table VI shows the results of estimated costs for several purchasing policies based on purchasing 5-day (Monday or Thursday departure) round-trip itineraries from NYC to MSP (∼265 simulated purchases in each test set). The table shows how costs vary based on preferences such as a customer requiring a nonstop itinerary.

5.1.1. Deterministic Feature Selection. The naïve strategy, called *earliest purchase*, purchases a ticket once for each day in the α day range. Its purchase episodes terminate with a purchase event on the first day of the episode. Its mean cost is equal to the mean of prices across the α day period. The lowest achievable cost, called the *optimal cost* strategy, is based on purchasing for each of the α episodes at the lowest price between the beginning of the episode and its departure date (denoted as 0% above optimal). The comparison methodology involving simulated purchases is similar to that used in Etzioni et al. [2003]. The action policy from Bing Travel's purchase recommender applied to the test set achieves a PAO 1% to 3% closer to optimal than

Table VI. Model Results Comparison on a Single Route for the Lowest Cost Itinerary on any Airline

Feature Selection Type	Learning Method	Learning Method Output	NYC-MSP Mon-Fri 0,1,2 stops	NYC-MSP M-F nonstop	NYC-MSP Tu-Th 0,1,2 stops	NYC-MSP Tu-Th nonstop
			(mean cost (in $), percentage above optimal (PAO, in %))			
Deterministic feature selection	Earliest Purchase	Buy/Wait	(317, 18.2)	(414, 21.4)	(309, 17.5)	(374, 24.2)
	Optimal Cost	Buy/Wait	(268, 0.0)	(341, 0.0)	(263, 0.0)	(301, 0.0)
	Bing Travel	Buy/Wait	(308, 14.9)	N/A	(306, 16.3)	N/A
	PLS w/ Most Recent Obs.	Regression	(314, 17.1)	(384, 12.6)	(294, 11.8)	(354, 17.6)
	PLS w/ Full Lag Scheme	Regression	(300, 11.9)	(398, 16.7)	(316, 20.1)	(345, 14.6)
Off-the-shelf methods	PLS w/ CFS	Regression	(313, 16.8)	(413, 21.1)	(308, 17.1)	(371, 23.2)
	PLS w/ BFS	Regression	(317, 18.2)	(416, 22.0)	(310, 17.8)	(369, 22.6)
Developer-guided feature selection	REPTree in Classification	Buy/Wait	(288, 7.4)	(388, 13.8)	(289, 9.9)	(382, 26.9)
	REPTree in Regression	Regression	(284, 5.9)	(375, 10.0)	(280, 6.5)	(334, 11.0)
	Ridge Regression	Regression	(293, 9.3)	(383, 12.3)	(316, 20.1)	(372, 23.6)
	nu-SVR	Regression	(295, 10.1)	(396, 16.1)	(289, 9.9)	(338, 12.3)
	PLS Regression	Regression	(280, 4.5)	(365, 7.0)	(276, 4.9)	(330, 9.6)

Note: All itineraries are 5 days (Monday to Friday, or Thursday to Tuesday). Cities are NYC (New York City) and MSP (Minneapolis, MN, USA). Note: "PAO" (% above optimal policy cost) for each method is computed as: $([\text{Score}_{method} - \text{Score}_{\text{optimal}}]/\text{score}_{\text{optimal}})$ so that optimal measures = 0%.

earliest purchase. Since Bing Travel does not separately predict for nonstop flights, there are no results for the nonstop category.

For deterministic feature-selection approaches, there are several benchmarks. The *Most Recent Observation* with PLS regression contains only the most recent value from each raw feature (i.e., no lagged variables). This method cannot leverage trends but can observe competitive relationships. The *Full Lag Scheme* with PLS regression contains all lags $\{0, -1, \ldots, -7\}$ from every feature. The method can predict trends and can leverage competitive relationships, but prediction performance may suffer due to the large number of features.

5.1.2. Off-the-Shelf Methods. *PLS with CFS* is a filter-based automated feature-selection method, coupled to PLS regression after the feature selection step. Its performance is worse than the performance of PLS regression alone (i.e., with no feature selection). The correlation between the target variable and individual features may not provide a predictive feature set in aggregate.

PLS with BFS is best-first search, a wrapper-based automated feature selection method utilizing greedy search, coupled to PLS regression for each feature set evaluation. The results are worse than earliest purchase. The approach may be poor due to the large search space caused by many raw features.

5.1.3. Developer-Guided Feature Selection Methods. Developer-Guided Feature Selection is coupled to five different learning algorithms using the feature classes and constraints given earlier. These experiments enumerate all valid feature sets given the constraints. The scores are for the best model found on the calibration set, which is then applied to the test set. The decision-tree algorithm, *REPTree*, is used in both classification mode

Table VII. Percentage Above Optimal Cost (PAO, as %) for Various Decision Theoretic Approaches Tested on the 7 (Domestic and International) Routes Shown (Using the Airport Codes) for a 5-Day Round Trip for Monday and Thursday Departures

			Model			
	Optimal	Our Model	Linear (LR)	Ripper	Earliest Purchase	Latest Purchase
Method Origin:	Baseline		[Etzioni et al. 2003]		Baseline	Baseline
HOU → NYC	0.0	4.3	13.9	19.8	14.8	53.9
MSP → NYC	0.0	3.9	14.5	21.7	14.3	48.8
NYC → CDG*	0.0	5.5	14.5	26.1	16.6	59.0
NYC → CHI	0.0	6.8	15.7	48.1	28.4	108.5
NYC → HKG*	0.0	6.4	13.3	36.0	17.3	39.0
NYC → MSP	0.0	5.2	14.0	33.0	18.5	58.1
SEA → IAD	0.0	4.5	14.2	18.8	12.8	43.3
Mean PAO	0.0	5.3	14.3	28.9	17.4	58.7
Savings margin (%)	11.0	7.25	0.514	−10.4	0.0	−32.3

*Denotes an international route.
Note: "PAO" is computed as a proportion: $(\text{cost} - \text{cost}_{\text{optimal}}) \div (\text{cost}_{\text{optimal}})$.) Cities are: HOU = Houston, TX, USA; MSP = Minneapolis, MN, USA; NYC = New York, USA; CDG = Paris, France; CHI = Chicago, IL, USA; HKG = Hong Kong, China; SEA = Seattle, WA, USA; IAD = Washington, DC, USA. Savings margin computed as % of earliest purchase cost and represents the savings a strategic consumer may experience using the model.

(to produce buy/wait signals directly) and regression mode (to build price predictions that are fed to the decision-threshold function). *REPTree in regression* performs well. It implicitly performs a feature-selection process when building the tree so that it may be able to prevent overfitting due to a large number of collinear features.

Ridge regression is a regularized linear regression algorithm used in multivariate domains with many (possibly collinear) variables. It may perform poorly due to overfitting because the number of observations (≈100) is insufficient relative to the number of features (≈1000). This makes overfitting more likely.

nu-SVR, a support vector machine method (with linear kernel), performs almost as well as PLS regression for this domain, but it is much more computationally expensive due to the large number of features for some lag scheme configurations.

PLS regression performs the best in this application within ∼6.5% of optimal cost on average from the targets in Table VI. The method implicitly does a dimensionality reduction on the training-set concept during calibration, thus it is more robust to overfitting due to the small number of training samples and large number of features.

Table VII shows that the optimal policy provides, on average, a 11.0% savings over earliest purchase. We denote this percentage as the *savings margin*. Our method of a lag scheme search coupled with PLS regression and a decision threshold achieves consistently closer to the optimal action sequence than any of the other methods compared. The PAO of ∼6.5% achieved by PLS (in Table VI) represents a savings of 8% over the earliest purchase strategy for the NYC→MSP route.

5.2. Bing Travel Performance Comparison

Bing Travel provides a prediction about whether or not the lowest cost ticket from any airline will be lower over the next 7 days. Even though Bing Travel has different and broader objectives than our study, it is surprising that it does not achieve a greater savings margin on the Any Airline target. We believe that this is due to its more risk-averse approach, which makes it significantly more likely than our method to advise immediate purchase.

This assertion can be validated by looking at the distribution of buy and wait signals computed for each day by the various policy generators: in the NYC→MSP M-F route, the optimal policy has only a 15% proportion of buy signals. It is noteworthy that the

best models constructed with our method emit a similar proportion of wait signals: in the NYC→MSP M-F route, the model with the lowest average cost ($280) emits a buy signal on only 34% of the days. Bing's model has a much higher proportion of buy signals: in the same route, the Bing model emits a buy signal 83% of the days. The results are similar for all routes and dates in our dataset: Bing emits buy signals for at least 70% of the days.

5.3. Multiroute Comparison

To show that the proposed method is generalizable to other routes (including international routes), we provide performance statistics on 7 routes in Table VII. The proposed method achieves an average of 69% of the optimal savings, which represents an average cost savings of 7.25% when compared to the earliest purchase strategy. Given the high cost of airline tickets, this represents a significant savings. For the purposes of comparison with existing approaches, we provide results of two decision theoretic methods from Etzioni et al. [2003]: Ripper and LR (an MA(1) time-series model in the notation of Box et al. [2013]). Those models use a smaller number of features compared to our model and do not leverage the competitive relationships between airlines when making predictions. We believe that predictions are improved by considering price competition between airlines.

5.4. Specific Preference Models

So far, we have shown how our model predicts the expected minimum price of all available flights on a specific route and departure date. We now show how our model can also predict prices of flights with specific desirable properties, such as flights from a specific airline, nonstop-only flights, or multisegment flights. Buyers are likely to have preferences about airline tickets beyond price, such as loyalty to a specific airline or the desire for overall minimum travel time. By comparing models with different target properties, buyers can determine the likely cost of their preferences.

A model for a specific preference (e.g., nonstop-only flights, flights with take-off time before 11 a.m., or flights from a specific airline only) can be generated by computing the correct price prediction target variable to train the regression model. The regression model's target price for each observation day will depend on the exact flight preferences. For example, the future minimum price of any airline and any number of stops (ALL-min-A) for the departure date will always be equal to or less than the future minimum price for a specific airline and nonstop-only flights (e.g., DL-min-0). In the terminology of machine learning, this corresponds to computing the label for each row in the dataset. The target label values will depend on the concept being learned. These values are computed from the subset of the airline itineraries that match the preference specified.

The effect of more specific preferences can be observed in the feature set configurations that are generated for each preference. In Table VIII, the legacy airlines (DL and CO) offering multiple types, including nonstop and multistop flights, may have different lag schemes based on the preference. The general pattern observed is that more specific preferences (such as nonstop-only flights) require more information in terms of more specific feature classes and more time lags than less specific preferences (such as any flight from any airline). Also, more desirable flights such as nonstop-only flights will tend to be more expensive than multistop ones. This is also observed in the lag scheme plots: our model is able to obtain an average price of $452 for Continental (CO) nonstop flights, but can obtain an average price of $345 for flights possibly with multiple stops.

For the LCAs, the lag schemes suggest that there are delays in each airline's response to changes in prices. For example, the LCA models use little (AirTran) or no (Frontier and Sun Country) information from the current day (lag offset 0) and previous day (lag

Table VIII. Optimal Lag Schemes for Specific Preferences of Airline and Number of Stops

Legacy Airlines

Continental Airlines (CO) — Nonstop

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A	•	•	•	•	•	•	•	•
ALL-S	•	•	•	•	•	•		
EACH-A		•	•	•	•	•		
EACH-S				•				

(MC: $452.7, EP: $483.6, PAO: 7.0%)

Continental Airlines (CO) — 0, 1, or 2-stops

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A	•	•	•	•	•	•		
ALL-S	•	•	•	•	•	•		
EACH-A								
EACH-S								

(MC: $344.8, EP: $386.3, PAO: 4.4%)

Continental irlines (CO) — 1-stop

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A		•	•					
ALL-S		•	•					
EACH-A		•						
EACH-S								

(MC: $422.7, EP: $456.1, PAO: 3.3%)

Delta Airlines (DL) — Nonstop

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A			•	•	•	•	•	•
ALL-S			•	•	•	•	•	•
EACH-A			•					
EACH-S			•					

(MC: $389.5, EP: $411.6, PAO: 7.1%)

Delta Airlines (DL) — 0, 1, or 2-stops

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A	•	•	•	•	•	•	•	•
ALL-S	•	•	•					
EACH-A		•						
EACH-S		•						

(MC: $366.8, EP: $393.8, PAO: 6.7%)

Low-Cost Airlines (LCAs)

Frontier (F9) — 0, 1, or 2-stops

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A			•	•	•	•	•	•
ALL-S				•	•	•	•	•
EACH-A						•	•	
EACH-S						•	•	

(MC: $319.0, EP: $357.2, PAO: 3.5%)

AirTran (FL) — 0, 1, or 2-stops

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A	•	•	•	•	•	•	•	•
ALL-S			•	•	•	•	•	•
EACH-A								•
EACH-S								

(MC: $315.0, EP: $335.7, PAO: 3.2%)

Sun Country (SY) — Nonstop

Class	Lagged Offsets 0	-1	-2	-3	-4	-5	-6	-7
DET	•							
ALL-A		•	•	•	•	•	•	
ALL-S		•						
EACH-A		•						
EACH-S								

(MC: $352.6, EP: $399.1, PAO: 1.3%)

Note: Experiments are for 5-day round trips departing on Thursdays. The statistics are as follows: "MC" refers to average model cost, "EP" refers to average earliest purchase cost, and "PAO" is the mean price of the model above the optimal policy price (expressed as %, closer to zero is better).

offset −1) for prediction. An alternative explanation for no current-day information in the model may be in the airline's role as a price leader. If an airline is setting prices aggressively, the airline will not make rapid changes in response to the actions of other companies in the market.

Table VIII shows that some LCA models (Sun Country and AirTran) do not use information at the EACH-STOP level. This suggests that these airlines do not use detailed information from other airlines to determine their responses. The prices for these airlines generally are set equal to or just below the prices of the legacy airlines, thus detailed information about the legacy airlines' pricing for nonstop and multistop flights may not be needed for prediction. Also, because the pricing relationships for the LCAs is

Table IX. Developer-Guided Feature Selection Constraint Set Sensitivity Analysis

Learning Method	Feature Class Hierarchy	NYC-MSP Mon-Fri	NYC-MSP M-F nonstop	NYC-MSP Tu-Th	NYC-MSP Tu-Th nonstop
		(mean cost (in $), percent above optimal cost (PAO, in %))			
PLS Regression	All lags	(300, 11.9)	(398, 16.7)	(316, 20.2)	(345, 14.6)
	Most recent	(314, 17.2)	(384, 12.6)	(294, 11.8)	(354, 17.6)
	Constraint Set A	(280, 4.5)	(365, 7.0)	(276, 4.9)	(330, 9.6)
	Constraint Set B	(291, 8.6)	(393, 15.2)	(279, 6.1)	(330, 9.6)
	Constraint Set C	(297, 10.8)	(401, 17.6)	(280, 6.5)	(353, 17.3)
	Constraint Set D	(292, 9.0)	(407, 19.4)	(280, 6.5)	(348, 15.6)

less sophisticated than the relationships of the legacy airlines, the prediction models should be more effective for predicting LCA pricing than for legacy airline pricing. This can be observed in the PAO values observed for LCAs compared to legacy airlines.

Using these statistics, it is also possible to reason about the relative costs of various preferences: for example, what is the expected price difference between a nonstop Delta flight and a flight on any available airline? Such information could help customers to quantify the expected costs of their preferences.

5.5. Feature Class Hierarchy Sensitivity Analysis

The principal benefit of Developer-Guided Feature Selection is from a reduction in the feature selection search space through the use of the practitioner-provided domain knowledge. This section considers how sensitive the results are to the constraint set choice. Experiments in Table IX examine the performance of alternative constraint sets. The original constraint set presented in Figure 6 is labeled as "Constraint Set A"; these results are consistent with results in Table VII. We modify the constraint set by swapping the location of ALL-STOPS and EACH-AGGREGATE; this is labeled as "Constraint Set B." Another modification increases the range of time-lagged variables from [0, −7] to [0, −10] time units, labeled as "Constraint Set C." This greatly increases the configuration search space. "Constraint Set D" involves a search with no constraints in the feature class hierarchy: all nodes are at the same level in the tree and there are no subset constraint relationships. The no-constraints search space is considerably larger, involving 1,414,562 possible configurations.

These experiments find that a different constraint set does not dramatically change the test set scores (A vs. B). Increasing the range of available lags beyond a 7-day pattern does not improve results (C). We conjecture that all constraint sets (A, B, C, and D) improve upon the performance of the no-feature-selection version because the feature-selection process reduces the number of features used in model calibration. This is especially beneficial for domains in which the number of observations (n) is small relative to the number of features (k): $n < k$. The exact choice of constraints is not critical. Of primary importance is the need to reduce the feature sets evaluated.

Regarding the performance differences between the constraint sets, we conjecture that the performance degradation observed by Constraint Set D is due to oversearching, the phenomenon of a search process exploiting transient patterns in the dataset that do not generalize well [Quinlan and Cameron-Jones 1995]. As the number of configurations evaluated increases, so does the likelihood of oversearching. Constraint Sets A, B, and C explore different (and much smaller) subsets of the search space than D. Constraint Set A is the one most congruent with our understanding of the variable relationships in this domain; we find its performance to be similar to or superior to other configurations.

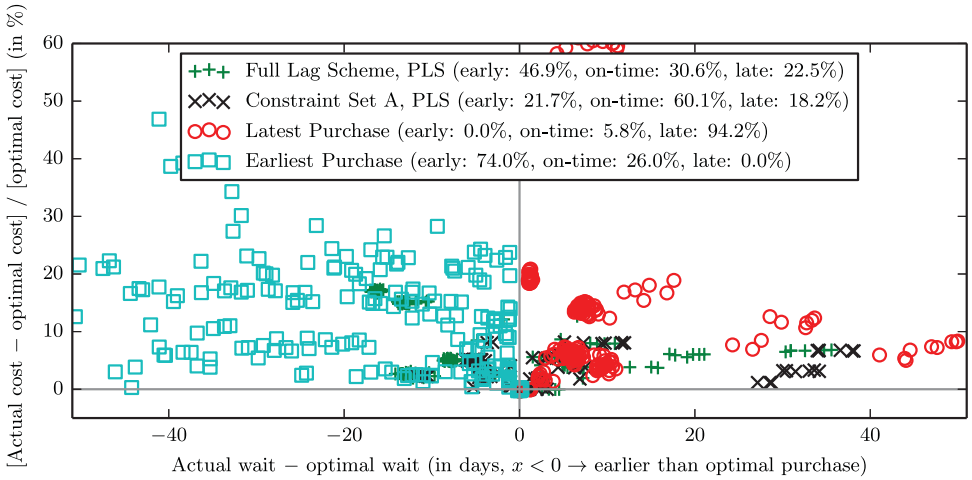

Fig. 7. Temporal accuracy of purchasing signals for 4 purchase-policy generators. Each marker corresponds to an individual purchase. Points with $x < 0$ indicate purchases that were before the optimal purchase time (categorized as "early") while points with $x > 0$ indicate purchase signals that waited more than the optimal number of days (categorized as "late"). The y-axis value indicates how much higher the purchase cost was than the optimal cost (denoted as "PAO," percentage above optimal). Gaussian noise is added to help visualize overlapping markers.

Table X. Statistics for Different Purchase Policy Generators

Method:	Full Lag Scheme			Constraint Set A			Latest Purchase			Earliest Purchase		
Lateness Bin	Early	On-time	Late	Early	On-time	Late	Early	On-time	Late	Early	On-time	Late
Bin Percentage (%)	46.9	30.6	22.5	21.7	60.1	18.2	0.0	5.8	94.2	74.0	26.0	0.0
Mean Lateness (in days)	-10.9	0.0	9.3	-5.0	0.0	13.6	—	0.0	10.1	-17.7	0.0	—
PAO (% above optimal)	9.2	0.0	4.3	4.1	0.0	4.0	—	0.0	15.8	13.6	0.0	—

5.6. Prediction Accuracy of Purchase Timing

Another way to analyze and compare the action signals from the methods presented is to consider the temporal accuracy of the purchase timing for each simulated passenger. Because the exact sequence of future prices is known for each simulated passenger, the exact number of days to wait to achieve the optimal (lowest) purchase cost is known. We can compare the number of days the algorithm emitted a wait signal against the optimal number of days to wait. This comparison is shown visually for the test set observations in Figure 7. The methods that emit markers closer to the origin are best because they are emitting a sequence of buy–wait signals closes to the optimal policy. In practice, this is difficult to achieve.

We show the results of earliest purchase (which always emits a buy signal) and the results of latest purchase (which emits a wait signal until the last possible day) as benchmarks. As a simple check, we can see, using the earliest purchase results, that the best price is achieved using an always-buy signal only 26.0% of the time. Also, waiting until the last possible time to buy (an always-wait signal) achieves the lowest possible cost only 5.8% of the time. Both of these approaches increase the overall cost above the optimal policy by (∼15%) as shown in Table X.

Table XI. Effect of Market Competition on Model Performance

Airline Route	No. of Significant Competitors	Mean Passengers per Day	Competition (HHI)	Earliest Purchase Std. Dev. (as %)	Our Model Savings Margin (as %)
NYC-CHI	9	6686	1861	6.79	12.834
MSP-NYC	6	1597	3228	5.65	8.958
NYC-MSP	7	1624	3296	5.60	7.091
HOU-NYC	5	2305	4484	5.53	6.615
SEA-IAD	2	385	7790	4.55	7.056

Note: Market competition statistics can be generated for US domestic routes using data from U.S. Department of Transportation [2012]. Routes are ordered by decreasing competition (increasing HHI value).

The two complex methods that we show on the graph are "Full Lag Scheme, PLS," which is the results using our decision-policy framework but without any feature selection search. As this method uses all the variables and time lags available, we expect that the price predictions may be more sensitive to movements of some variables. In the time accuracy plot, this method is much more likely to buy before the optimal purchase timing (46.9% of the purchases are before the optimal purchase timing). The purchase timing results of the best method based on our experiments is shown as "Constraint Set A, PLS". It is able to achieve the optimal purchase timing on 60.1% of all simulated purchases. It reduces the proportion of both late and early purchase signals. Also, it is possible to see from the markers in Figure 7, even for late and early signals, that the prices experienced by the best method are closer to the optimal price.

5.7. Market Competition

Varying market competition is one potential explanation for the differences in performance across routes. We examine this in more detail by computing market competition for each route and plotting it with the savings margin. To measure market competitiveness, we use the Herfindahl-Hirschman Index (HHI), an economics-based measure of market consolidation [Rhoades 1993]. The index is computed on the market share percentages of all participants and ranges from almost zero to 10,000 (perfect monopoly):

$$\text{HHI} = \sum_i \left(m_i{}^2\right), \tag{4}$$

where m_i is the percentage market share of participant i.

Markets having an HHI value greater than 2,500 are considered to be very noncompetitive. Market share information is computed for US domestic airline routes from a dataset containing a sample of all domestic airline tickets issued in quarter 3 of 2011 [U.S. Department of Transportation 2012].

Some routes are naturally more competitive than others due to the number of airlines that have market power and price aggressively to affect demand. Airlines that actively serve a route are ones that handle a significant volume of the route's traffic. We define a *significant competitor* on a route as an airline serving more than 1% of all passengers. By comparing the results of our proposed prediction model across routes, it is possible to see the effects of competition on the model savings. Table XI reveals that, for the most competitive routes (e.g., NYC-CHI, MSP-NYC), the savings margin experienced by the model is larger at around 10%. For routes that are less competitive and have a larger HHI value (e.g., SEA-IAD, HOU-NYC), the savings margin is reduced. Competition is also somewhat correlated with market size (as measured by mean passengers per day), but our results find that competition is a better predictor.

Another correlated measure is seen by looking at the variance of the prices from the earliest purchase strategy as measured using standard deviation in Table XI. The lower variance of daily minimum prices shown in the NYC-CHI route is likely due to the large number of competitive carriers along the route and the large number of passengers. In contrast, the SEA-IAD route has fewer "significant competitors," thus individual airlines can assert greater pricing power. An analysis in Vowles [2000] confirms a similar behavior of dominant competitors using a regression analysis of fares on US domestic routes.

6. CONCLUSIONS AND FUTURE WORK

We have presented a method for predicting airline ticket prices. The method uses historical data from which relevant features are extracted to build a predictive model of prices for all the airlines serving an origin–destination pair of airports on a specific departure date. Our results show that, given publicly observable historical data, airline ticket prices can be predicted. While buying at the earliest opportunity is the most obvious purchase policy, we show that is not the best policy in most cases. The long lead-time price may not be the lowest price available for that flight. There is also an opportunity cost associated with early commitment: a customer risks being locked into a specific schedule that may need to be changed (for a fee).

Because there is sufficient structured price volatility on many airline routes, there are significant opportunities for savings. To our knowledge, our results represent the state-of-the-art in airline ticket price prediction using consumer-accessible data. We believe that there is a significant market for these kinds of models in the hands of consumers not only to reduce their travel costs, but also to determine the range of expected prices for an itinerary, and to quantify the cost of airline and routing preferences.

The main novelty of our approach lies in our Developer-Guided Feature Selection technique, which captures temporal dependencies in the data via time-delayed features, and which reduces the number of features by using a class hierarchy among the domain features and pruning them based on in-situ performance.

Developer-Guided Feature Selection has wide applicability to other multivariate domains in which basic domain knowledge is common but not utilized and for whcih there are significant intravariable and intervariable temporal relationships. Building the feature class hierarchy requires only basic domain knowledge to be successful; greater expertise in hierarchy construction could improve efficiency even more. Feature selection contributes to the prevention of overfitting (evident from the poor performance of off-the-shelf feature-selection approaches), which can occur when there are many features and few training instances. In addition, it enables the discovery of meaningful relationships among features and facilitates domain understanding. By examining the relative performance of candidate lag schemes, domain knowledge can be extracted: the significance of individual features can be determined by observing their presence (or absence) in the calibrated lag scheme and feature set.

Dynamic pricing could be beneficial in other types of industries. For instance, industries that store inventory traditionally have concentrated their efforts on tracking inventories to reduce inventory size because of the high cost of changing prices [Elmaghraby and Keskinocak 2003]. However, the availability of demand data, the ability to inexpensively change prices, and the use of decision-support tools can change the situation and bring dynamic pricing to industries that currently rely on inventory control (see Groves and Gini [2013a] for an example in simulated supply-chain management).

From the consumer perspective, optimizing purchase timing is beneficial in other markets as well. While consumer goods markets are clearly different from the airline ticket market, there are similar systematic behaviors that could be leveraged through

prediction [Agrawal et al. 2011]. In auction markets, such as the eBay online marketplace, properties of auctions, such as ending hour, seller feedback rating, and the like, have been shown to affect the final auction prices [Bajari and Hortacsu 2003; Lucking-Reiley et al. 2007]. Such information has been successfully applied in auction arbitrage [Raykhel and Ventura 2009].

APPENDIX

A. COUNTING VALID CONFIGURATIONS GIVEN CONSTRAINTS

The number of configurations for a given a set of constraints can be computed precisely, and does not require a complete enumeration. For a chain of subset constraints among feature classes that allow contiguous subsets of lags and *null*, the number of possible configurations can be computed with the following recurrence relations:

$$A(k,n) = \begin{cases} k & \text{if } n = 1 \\ n & \text{if } k = 1 \text{ and } n > 1 \\ \binom{n+(k+2)}{n-1} + A(k-1,n) & \text{otherwise} \end{cases} \tag{5}$$

$$B(k,n) = \begin{cases} 1+k & \text{if } n = 1 \\ B(k,n-1) + A(k,n) & \text{if } n > 1, \end{cases} \tag{6}$$

where k corresponds to the number of feature-class levels, and n corresponds to the number of unique lag values for each feature class. For a chain of length four with 8 possible values $\{0, 1, 2, \ldots, 7\}$ and null for each feature class, the number of configurations is the value of $B(4, 8) - 1$, where minus one is due to removing the assignment with $\varnothing$ (empty set) values for all feature classes. The result is $(8518 - 1) = 8517$.

REFERENCES

Rakesh Agrawal, Samuel Ieong, and Raja Velu. 2011. Timing when to buy. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 709–718.

Enrico Bachis and Claudio A. Piga. 2011. Low-cost airlines and online price dispersion. *International Journal of Industrial Organization* 29, 6, 655–667.

Patrick Bajari and Ali Hortacsu. 2003. The winner's curse, reserve prices, and endogenous entry: empirical insights from eBay auctions. *RAND Journal of Economics* 34, 2, 329–355.

Peter P. Belobaba. 1987. Airline yield management. An overview of seat inventory control. *Transportation Science* 21, 2, 63–73.

George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. 2013. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, Hoboken, NJ.

Sijmen de Jong. 1993. SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems* 18, 3, 251–263.

Ke-Lin Du and M. N. S. Swamy. 2014. Recurrent neural networks. In *Neural Networks and Statistical Learning*. Springer, London, 337–353. DOI: http://dx.doi.org/10.1007/978-1-4471-5571-3_11

Wedad Elmaghraby and Pinar Keskinocak. 2003. Dynamic pricing in the presence of inventory considerations: Research overview, current practices, and future directions. *Management Science* 49, 10, 1287–1309.

Oren Etzioni, Rattapoom Tuchinda, Craig Knoblock, and Alexander Yates. 2003. To buy or not to buy: Mining airfare data to minimize ticket purchase price. In *SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 119–128.

Graham Francis, Ian Humphreys, Stephen Ison, and Michelle Aicken. 2006. Where next for low cost airlines? A spatial and temporal comparative study. *Journal of Transport Geography* 14, 2, 83–94.

William Groves. 2013. Using domain knowledge to systematically guide feature selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, Palo Alto, CA, 3215–3216.

William Groves and Maria Gini. 2013a. Improving prediction in TAC SCM by integrating multivariate and temporal aspects via PLS regression. In *Agent-Mediated Electronic Commerce. Designing Trading*

Strategies and Mechanisms for Electronic Markets. *Lecture Notes in Business Information Processing*, Vol. 119. Springer, Berlin, 28–43.

William Groves and Maria Gini. 2013b. Optimal airline ticket purchasing using automated user-guided feature selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, Palo Alto, CA, 150–156.

Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *The Journal of Machine Learning Research* 3, 1157–1182.

Mark A. Hall. 1999. *Correlation-Based Feature Selection for Machine Learning*. Ph.D. Dissertation. The University of Waikato, Waikato, New Zealand.

Mark A. Hall. 2000. Correlation-based feature selection for discrete and numeric class machine learning. In *International Conference on Machine Learning (ICML)*. Morgan Kaufmann, San Francisco, CA, 359–366.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer, New York.

Arthur E. Hoerl and Robert W. Kennard. 2000. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics* 42, 1, 80–86.

Ian T. Jolliffe. 1982. A note on the use of principal components in regression. *Journal of the Royal Statistical Society (Applied Statistics)* 31, 3, 300–303.

Ron Kohavi and George H. John. 1997. Wrappers for feature subset selection. *Artificial Intelligence* 97, 1–2, 273–324.

Yuri Levin, Jeff McGill, and Mikhail Nediak. 2009. Dynamic pricing in the presence of strategic consumers and oligopolistic competition. *Management Science* 55, 1, 32–46.

David Lucking-Reiley, Doug Bryan, Naghi Prasad, and Daniel Reeves. 2007. Pennies from Ebay: The determinants of price in online auctions. *The Journal of Industrial Economics* 55, 2, 223–233.

Benny Mantin and David Gillen. 2011. The hidden information content of price movements. *European Journal of Operational Research* 211, 2, 385–393.

Benny Mantin and Bonwoo Koo. 2010. Weekend effect in airfare pricing. *Journal of Air Transport Management* 16, 1, 48–50.

Harold Martens and Tormod Næs. 1992. *Multivariate Calibration*. John Wiley & Sons, Hoboken, NJ.

Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. 2002. Feature selection algorithms: a survey and experimental evaluation. In *IEEE International Conference on Data Mining (ICDM)*. IEEE, Piscataway, NJ, 306–313.

K. Obeng and R. Sakano. 2012. Airline fare and seat management strategies with demand dependency. *Journal of Air Transport Management* 24, 42–48.

Claudio A. Piga and Nicola Filippi. 2002. Booking and flying with low-cost airlines. *International Journal of Tourism Research* 4, 3, 237–249.

Steven L. Puller and Lisa M. Taylor. 2012. Price discrimination by day-of-week of purchase: Evidence from the U.S. airline industry. *Journal of Economic Behavior & Organization* 84, 3, 801–812.

J. R. Quinlan and R. M. Cameron-Jones. 1995. Oversearching and layered search in empirical learning. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, San Francisco, CA, 1019–1024.

Ilya Raykhel and Dan Ventura. 2009. Real-time automatic price prediction for eBay online trading. In *Proceedings of the Innovative Applications of Artificial Intelligence Conference.* AAAI, Palo Alto, CA, 135–140.

Stephen A. Rhoades. 1993. Herfindahl-Hirschman index, the. *Federal Reserve Bulletin* 79, 188–189.

Tim Sauer. 1994. Time series prediction by using delay coordinate embedding. In *Time Series Prediction: Forecasting the Future and Understanding the Past*, Andreas S. Weigend and Neil A. Gershenfeld (Eds.). Addison Wesley, Boston, MA, 175–194.

Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. 2000. New support vector algorithms. *Neural Computation* 12, 5, 1207–1245.

Barry C. Smith, John F. Leimkuhler, and Ross M. Darrow. 1992. Yield management at American Airlines. *Interfaces* 22, 1, 8–31.

Janakiram Subramanian, Shaler Stidham Jr., and Conrad J. Lautenbacher. 1999. Airline yield management with overbooking, cancellations, and no-shows. *Transportation Science* 33, 2, 147–167.

U.S. Department of Transportation. 2012. Origin-Destination Survey. Bureau of Transportation Services.

Timothy M. Vowles. 2000. The effect of low fare air carriers on airfares in the US. *Journal of Transport Geography* 8, 2, 121–128.

Eric A. Wan. 1994. Time series prediction by using a connectionist network with internal delay lines. In *Time Series Prediction: Forecasting the Future and Understanding the Past*, Andreas S. Weigend and Neil A. Gershenfeld (Eds.). Addison Wesley, Boston, MA, 195–218.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA.

Svante Wold, Harold Martens, and H. Wold. 1983. Multivariate calibration problem in chemistry solved by the PLS method. *Matrix Pencils* 973, 18, 286–293.

Zheng Alan Zhao and Huan Liu. 2011. *Spectral Feature Selection for Data Mining*. Chapman & Hall/CRC, London, UK.

Yun Zhou, Norman Fenton, and Martin Neil. 2014. Bayesian network approach to multinomial parameter learning using data and expert judgments. *International Journal of Approximate Reasoning* 55, 5, 1252–1268.

Received May 2014; revised December 2014; accepted February 2015