# Econometric Institute Report

EI 2006-27

Erasmus School of Economics
Erasmus University Rotterdam
The Netherlands

# A Support System for Predicting eBay End Prices

Dennis van Heijst Rob Potharst

Michiel van Wezel *

Econometric Institute,

Erasmus School of Economics Erasmus University

P.O. Box 1738, 3000 DR, Rotterdam, The Netherlands

October 25, 2006

**Abstract**

We create a support system for predicting end prices on eBay. The end price predictions are based on the item descriptions found in the item listings of eBay, and on some numerical item features. The system uses text mining and boosting algorithms from the field of machine learning. Our system substantially outperforms the naive method of predicting the category mean price. Moreover, interpretation of the model enables us to identify influential terms in the item descriptions and shows that the item description is more influential than the seller feedback rating, which was shown to be influential in earlier studies.

**Keywords: boosting, eBay, electronic auctions, text mining**

*Corresponding author. Phone: +31 10 4081339. Fax +31 10 4089167. E-mail: mvanwezel@few.eur.nl

# 1 Introduction

Online auctions are hot. The world's largest online auction site eBay reports in its first quarter financial report over 2006 a net revenue of $1.390 billion, realizing a growth rate of 35% in consecutive years [10]. For researchers with an interest in data mining, online auctions offer the opportunity to collect and mine large data sets at low costs.

The market price of a product is generally non-stationary at eBay – it fluctuates over time. It is even possible that identical items receive different bids at any given point in time. Merchants might buy item at eBay and try to re-sell these items with a profit. The success of these merchants depends on their ability to find bargains and, of course, their bidding strategy. Finding these bargains can be made easier by using a support system.

A recent paper [15] introduces the 'Auction Advisor' system, which simplifies the search for bargains by presenting its user with relevant information like the current bid and a recommended price based on recently closed auctions. Using this standardized presentation, the user is able to make bidding decisions within a short amount of time. In this paper we improve this recommended price by making a price prediction based on several relevant characteristics of the auction: the number of pictures, the feedback rating and the description of the item.

A substantial amount of research has been carried out on the analysis of historical auctions using data mining and statistical techniques. (An interesting review of this work from an economics perspective is given in [5].) Most of this work focuses on finding factors determining the auction end price. Reference [9] for example, tries to find such factors using a data set of ancient coin sales at eBay and finds that the number of participants, the use of reserve prices, and seller reputations are determinants of the end price.

Others aim at characteristic behavior like last moment bidding [25]. For only a few studies the prediction of auction prices is the central problem. Several data mining methods are compared in [14] in order to find the most suitable method for price predictions, while [29] constructs a dynamical forecasting model, which can update the predicted price of an ongoing auction based on newly arrived information.

Unlike previous studies, we incorporate the textual information contained in the item description in our system when predicting the auction end price. To this end, our system downloads data on a large number of closed auctions from the eBay site. These data are then pre-processed and fed to a price-prediction model. Section 2 below discusses the data collection system and the pre-processing steps. The price-prediction model makes use of a vector space representation of the descriptions of the items [26]. Each position in the vector represents the occurrence-count of a specific word in an item description. This representation, known as the bag-of-words representation, is often used in Information Retrieval Systems. In these systems, the distance between the bag-of-word vectors of strings is used to find similar strings or documents.

Instead of using similarity calculations, our price-prediction model is based on boosting [13]. Boosting creates an ensemble of models that collectively make a prediction, in our case for the end price of the auction. We use decision trees as the individual models that form the ensemble, as is often done. Decision trees select important input dimensions in the course of their calibration process. This is a desirable property in text mining, as the number of input dimensions is very high. Section 3 discusses the models that we use in our system in more detail.

We test our system in two experiments described in Section 4. The paper

ends with conclusions and a discussion in Section 5.

## 2 Data

Collecting data from eBay can be done using a web crawler or using eBay's API (Application Programmers Interface). Our system uses a web crawler since the API allows only a limited number of calls per day. The crawler was written in Java. For our experiments, this web crawler was programmed to download closed auction pages at eBay during a period of one month, in August 2005. The downloaded auction pages are the main HTML pages from eBay, they do not include the seller's feedback pages nor do they include the bidding history. Figure 1 shows an example of (part of) a downloaded auction page.

[Figure 1 about here.]

### Features Included

The prediction module in our system does not deal with the unstructured HTML data downloaded by the crawler. Instead, it needs input features with clear semantics that are relevant for the auction end price. Our prediction model is based upon the following features: The *feedback rating*, the *number of pictures*, and the *description of an item*. First, we will summerize the reasons for using these features. Next, we will discuss how we construct these features from an eBay HTML page.

The first feature included is the feedback rating. EBay has a reputation system called the Feedback Forum. This system captures the reputation of an eBay member based on earlier transactions this member participated in. There are several studies indicating that there is a relationship between a

seller's reputation and the expected auction end-price, see, e.g., [4,23]. This is why we added the feedback score as an element in our price prediction model.

A feedback mechanism falls into the broader category of trust mechanisms in e-commerce, which has receives a lot of attention in the literature, see, e.g. [1–3, 16, 20]. Trust is not only created as a result of a high feedback rating. Bidders want to have as much information as possible about the item for sale. This information helps to reduce the uncertainty on the item's quality, rather than reducing uncertainty on the seller as reputation mechanisms do.

Some elements of information are difficult to describe in words. For example, a 'slightly worn shoe' could be seriously worn or be in near-mint condition. In such a case, pictures of the item provide bidders with useful and objective information. For that reason we included the number of pictures in an item description as a feature.

Another way for the seller to overcome the information asymmetry between the seller and the buyer is by describing the item in words. This description of an item is written in natural language. The text usually contains information like the possible uses of an item and the state of the item. It may also contain information on the transaction, such as return policies and shipping fares. Since this information is useful to the bidders the item description is included in the feature set.

Interestingly, it is a well-known result in auction theory [17] that the seller should disclose all information about the object being auctioned that is potentially of use to the bidders. This policy of making information publicly available raises the expected revenue for the seller. This is again a motivation for trying to capture the available information is the features of

our prediction module.

### Data Pre-processing

We now describe how the above mentioned features were constructed from a raw eBay HTML page, of which Figure 2 shows an example. Retrieving price and feedback rating was straightforward: they are parsed from the page by looking for the first occurrence of *Winning bid:* and *Feedback* respectively. The number of pictures can be found by counting the number of $<$ `img` $>$ tags in the description of the item.

[Figure 2 about here.]

The description of an item is written in natural language. Since the datamining method we use requires numerical input data, we need to transform the text of an item description into a numerical representation. This transformation is performed using a dictionary. Each individual item description is encoded by the frequency of occurrence of each word from the dictionary. This frequency can be either Boolean, indicating whether the word does or does not appear in the text, or numerical, indicating the number of occurrences in the text. This method is known as the bag-of-words (BOW) method [26, 30].

There has been some criticism on BOW. One argument is that when we use BOW, we lose semantics [19]. By looking at words only, we lose the extra information given in a sentence. For instance, adding the word *not* to a sentence may change its semantics completely. This change in meaning is not detectable by just increasing the occurrence of *not*, when using large texts. The key assumption of BOW, namely that the position of a word in a document does not matter, does not hold, but in practice BOW does perform

quite well [18, 27], and it is easy to implement. The following paragraphs illustrate how the dictionary of a description is made.

When creating the vectors we use a local pooled dictionary. This dictionary is therefore created by the words in the observed documents. Using these local pooled dictionaries over widely used ones is recommended in [30]. The authors of [30] conclude that the words in and the size of the dictionary are very important for the computational speed and the results of the prediction model.

To diminish the size of the dictionary we use Porter's stemming algorithm [21]. This algorithm strips the endings from many words in English. One example of this stripping is to convert plurals to their singular form by stripping the letter *s*. It is used as part of a term normalization process that is usually done when setting up Information Retrieval Systems. It assumes that although these words differ in quantity, they do belong to the same term. Therefore, we can change words to their root by stripping off the suffix, without changing what is meant. In Figure 3 you can read this paragraph stripped by Porter's stemming algorithm.

[Figure 3 about here.]

The frequency of occurrence of each word was counted after stemming the description. We introduced a lower bound, $f_l$, to filter out infrequently used words from our dictionary. This means that only words which have at least a total occurrence of $f_l$ were considered. Besides lowering the computational requirements of the systen, the motivation for this lower bound is that our prediction model needs a certain number of examples to be able to correctly recognize the importance of a word: The model could easily over-fit on a misspelled or infrequent word, because it lacks counter examples.

The over-fitting problem does not only occur with infrequent words, but also with words which occur too frequently. These words are referred to as stop words. Stop words are commonly used words like *I*, *is* and *and*. These words do not contribute to the information that a description conveys. Therefore, these words are filtered out. Our list of stop words was not generated from the local dictionary, but a predefined frequently used list was downloaded from the internet.[1]

[Figure 4 about here.]

A BOW representation is usually stored as a vector where each dimension holds the occurence or count of a specific word. An example of such a vector is shown in Figure 4. It shows the previously stemmed paragraph, after filtering it with our list of stop words. (In this example the lower bound $f_l$ was set to 1.) For use in the support system, we extended our BOW vectors with dimensions for the feedback rating, the number of pictures and the price, as was discussed earlier.

## Data sets used in the Experiments

We downloaded two independent data sets from two well defined auction categories over a period of 4 weeks in August 2005. These data sets were used to test our price prediction algorithm in a series of experiments described in Section 4.

The first data set concerns auctions of Canon digital cameras. We hypothesized that our learning algorithm would use technical terms in the item description among other terms as indicators for item value. Having these technical terms it would be able to classify cameras by type or model. We

[1]The url for this list is `http://www.lextek.com/manuals/onix/stopwords1.html`

should remark that the data set does not only contain cameras, but also contains accessories like lenses and batteries.

The second data set concerns auctions of Nike men's shoes. Prior to our experiment we expected the learning algorithm to classify auctions based on the terms 'used' and 'new'. Our algorithm should also be able to recognize models.

We used the search mechanism of eBay to select the auctions for our data sets. Unfortunately, the search mechanism also returned auctions outside our category. For example, our dataset included an auction of Nike shoes previously belonging to the famous basketball player Michael Jordan. Although these are Nike men's shoes, we did not want to predict collectibles. We therefore excluded outliers of this type by filtering them using a box-plot of the item's price. Both very large and very small values were excluded. Table 1 gives a summary of the data sets and the pre-processing parameters.

[Table 1 about here.]

## 3 Methodology for Price Prediction

This section discusses the machine learning techniques used in our system. We denote the available data by $D = \{(\vec{x}_i; y_i)\}_{i=1}^{N}$. An instance (observation, row) $(\vec{x}; y)$ consists of a vector of $J$ attribute values $\vec{x} = (x_1, \ldots, x_J)$ and a target value $y$. The $J$ attributes are the explanatory or independent variables, in our case the term counts and other input features cf. Section 2. The target is the explained or dependent variable, in our case the auction end-price.

### Classification and Regression Trees

CART (Classification And Regression Trees) [8] is one of the most frequently used methods for constructing decision trees. In this paper we use the CART regression tree.

A regression tree (see Figure 5 for an example) consists of *decision nodes* and *leaf nodes*. Each decision node has two child nodes, which may again be decision nodes or leaf nodes. The *root* of the tree is on the very top – it is the only node in the tree without an ancestor. Every decision node (also called non-terminal node) contains a split criterion, which divides the data at that point into two parts. This split criterion has the format $x_j \leq c_s$ for continuous variables, where $x_j$ is the $j^{th}$ variable and $c_s$ is a constant that may be different for each split $s$. For a categorical variable the split criterion looks like $x_j \in V$, with $V \subset W_j$. Here $W_j$ collection of all possible categories of variable $x_j$. The terminal nodes (leafs) contain a $\hat{y}$ value, an estimate for the target value in that leaf. In practice this value is taken to be the average of all observed $y$ values in that leaf.

The example regression tree shown in Figure 5 can be used for prediction as follows. Suppose a new vector $\vec{x}$ is presented to us, what will be our prediction of the target value for this vector? We begin at the root and if $\vec{x}$ satisfies the split criterion we turn left; if not we turn right. We keep on doing this until we reach a terminal node and use the $\hat{y}$ value in that node as our prediction of the target value for $\vec{x}$.

[Figure 5 about here.]

Decision trees are usually built in two phases. The first phase is a *growing* phase, the second phase is a *pruning* phase. In the growing phase, the tree is grown until error reduction on the training set is no longer possible or

a predetermined threshold has been reached. The resulting model usually overfits the data, and this is countered in a pruning phase, where the tree is shrunk until the error on a hold-out sample, the *pruning set*, is minimal. Details on the CART procedure for growing and pruning can be found in, e.g., [8,24]. Here, it is sufficient to remark that, given a data set $\{(\vec{x}_i; y_i)\}_1^N$, the CART algorithm constructs a regression tree $B$ that attempts to minimize the squared error loss

$$E_{\vec{x},y}(B(\vec{x}) - y)^2,$$

where $B(\vec{x})$ denotes the prediction of tree $B$ for input vector $\vec{x}$.

In the context of boosting, discussed below, the pruning phase of the decision tree algorithm is usually skipped and instead the tree size is limited to a predetermined depth. In the most extreme case the tree depth is 1. The tree then consists of a single decision node and two leaves. Such a special tree is called a *decision stump.* Although a single decision stump has very limited modeling power, an ensemble of such stumps is able to model complex relationships.

Regression trees have some advantages over the commonly used method of linear regression. In the first place, in contrast to regression functions, regression trees are able to determine themselves which of the attributes are to be used for modeling the relationship with the target variable. Another advantage is that regression trees are able to model interactions between attributes and non-linear relationships with the target, without a required explicit transformation of the inputs. Furthermore, in contrast to many parametric models, regression trees can handle categorical variables and missing values without transformation of the data.

A drawback of decision trees is their instability – the implemented model depends heavily on the exact data set used for model creation, and a small change in the data may have large consequences for the model. Ensemble methods, such as bagging [6] and boosting, have a stabilizing effect by averaging over a number of decision trees. We discuss boosting next.

### Boosting

Boosting is a method to combine multiple models to improve performance, i.e. to reduce the error on unseen data. Boosting was first applied to and developed for classification problems (with categorical response) in [11, 12]. In a classification context boosting seemed to be able to strongly reduce the error rate on out-of-sample data in many cases [7]. The idea behind boosting is to create a sequence of models, called base learners, in which each subsequent base learner focuses on the residual error of the previous base learners. Often, these base learners are decision trees or stumps.

The original Freund and Schapire boosting algorithm for classification, AdaBoost.M1, was only applicable to binary classification problems. For these problems, the model predicts whether an instance belongs to a class or not. The model thus has a 0/1 output and the quality of the model is measured with the $0-1$ loss function, which counts the number of misclassifications. For modeling eBay end-prices this loss function is not suitable. Instead, we need a regression loss function that measures the deviance between two numerical values, as usual in regression. This means that we cannot apply the AdaBoost.M1 algorithm to eBay end-prices. Instead, we use a special boosting algorithm, suitable for regression problems.

Various boosting algorithms have been designed for regression. Friedman [13] developed LSBoost, LADBoost and MBoost based on the squared,

absolute and Huber loss function respectively. (All these loss functions apply to regression problems.) In this paper we will use Friedman's LSBoost algorithm. This algorithm was chosen because, contrary to many other boosting algorithms, it has a solid mathematical foundation: it is an instantiation of a general boosting algorithm for general loss functions named GradientBoost.

We now give a brief description of GradientBoost and LSBoost. Contrary to fitting a single model, like the decision tree $B$ above, boosting starts with an initial guess $F_0$ and then fits a *sequence* of $M$ models $B_1, \ldots, B_M$ (the base learners) which are subsequently combined in a weighted manner. The final model is thus

$$F_M(\vec{x}) = F_0(\vec{x}) + \nu \sum_{m=1}^{M} \rho_m B_m(\vec{x}).$$

Here, $\rho_m$ denotes the weight for model $m$ and is determined by the algorithm. $M$, the number of iterations, is set by the user. The number $\nu$ with $0 < \nu \leq 1$ denotes a regularization parameter called the *learning rate*. Small values of $\nu$ will help prevent the algorithm to overfit the training data.

Note that at the $m^{th}$ iteration, $B_m()$ is added to $F_{m-1}$:

$$F_m(\vec{x}) = F_{m-1}(\vec{x}) + \nu \rho_m B_m(\vec{x}).$$

It makes sense to choose $B_m()$ such that it minimizes the residual error of $F_{m-1}$. Roughly speaking, given a general loss function $L(y, F)$, $B_m$ attempts to minimize the expected value of this loss function over the data set:

$$B_m = \arg\min_{B} \sum_{i=1}^{N} L(y_i, [F_{m-1}(\vec{x}_i) + B(\vec{x}_i)]). \tag{1}$$

In practice this is done by fitting *pseudo responses* $\tilde{y}_i$ in each iteration

$$B_m = \arg\min_B \sum_{i=1}^{N} \{\tilde{y}_i - B(\vec{x}_i)\}^2.$$

The values of the pseudo-responses depend upon the loss function in question. When GradientBoost is applied to the squared error function $L(y, F) = (y - F)^2/2$ that is common in regression, the pseudo-responses are given by $\tilde{y}_{i|m} = y_i - F_{m-1}(\vec{x}_i)$.

The minimization over $B$ in Equation 1 is performed by minimizing over $B$'s parameter space. If $B$ is a tree, these parameters are the split variables and split points in the decision nodes, and $B_m$ is the tree that gives the best fit of the $\tilde{y}$ values in iteration $m$. Figure 1 summarizes the LSBoost algorithm.

**Input**: data set with instances $\{\vec{x}_i; y_i\}_1^N$
number of iterations $M$
learning rate $\nu$
**Output**: Model $F(\vec{x})$
$F_0(\vec{x}) = \bar{y}$
**for** $m = 1$ *to* $M$ **do**
  $\{\tilde{y}_i = y_i - F_{m-1}(\vec{x}_i)\}_1^N$
  train $B_m(\vec{x})$ using $\{\vec{x}_i; \tilde{y}_i\}_1^N$
  $\rho_m = \arg\min_\rho \sum_{i=1}^{N} [\tilde{y}_i - \rho B_m(\vec{x}_i)]^2$
  $F_m(\vec{x}) = F_{m-1}(\vec{x}) + \nu\rho_m B_m(\vec{x})$
**end**

**Algorithm 1**: LSBoost algorithm [13].

## Interpretation

Parametric techniques often have the advantage that a useful interpretation can be given to the model parameters, e.g., in linear regression the model parameters can be interpreted as the weights of the item characteristics. Although not parametric, regression trees are also highly interpretable and

can be written as an equivalent set of if-then rules. Boosted trees lack both these appealing properties. Fortunately, at least to some degree boosted models can be interpreted by using *relative importance plots.*[2]

Relative importance plots visualize how important the various independent variables are relative to one another in predicting the dependent variable. Relative importance plots were developed for trees by Breiman et al. [8], but they are easily generalizable to an ensemble of trees. For a single CART model, the following formula measures the importance of variable $x_j$:

$$\hat{I}_j^2(B) = \sum_{n=1}^{K-1} \hat{i}_n^2 \chi(v_n = x_j) \ .$$

Here the summation is over the $K-1$ non-terminal nodes in tree $B$ having $K$ terminal nodes and $\chi()$ denotes the indicator function. $v_n$ is the split variable of node $n$. The factor $\hat{i}_n^2$ measures the improvement in squared error as a result of the split in node $n$, and can be computed as follows:

$$\hat{i}_n^2 = \frac{w_l w_r}{w_l + w_r}(\bar{y}_l - \bar{y}_r)^2 \ .$$

Here $w_l$ and $w_r$ are the probabilities an instance turns to the left or right child node of node $n$, $\bar{y}_l$ and $\bar{y}_r$ are the mean target values for both children. Both the probabilities and the means are computed on the training set and saved in the CART model.

To compute the $\hat{I}_j^2$'s of a boosting model it is sufficient to average the

[2]Another frequently used interpretation tool is the partial dependence plot, which graphically depicts the shape of the dependency of the target variable upon an input variable, see [13].

$\hat{I}_j^2$'s of the base learners:

$$\hat{I}_j^2 = \frac{1}{M} \sum_{m=1}^{M} \hat{I}_j^2(T_m) \ .$$

A variable $x_j$gets a high importance $I_j^2$ when it is used in many splits, but more importantly when it is used in splits that divide the data in two almost equally large parts with a large difference in mean target value, thus contributing a lot to the total error reduction.

Finally, the variable $x_j$ with the highest importance gets a relative importance index of $RI = 100$ and the other indices are adjusted to this:

$$RI_j = \frac{\hat{I}_j^2}{\hat{I}_{max}^2} 100 \ .$$

We will use relative importance plots below in Section 4 to identify the most important terms that influence prices in both the Nike and the Canon data sets.

# 4 Experiments & Results

We experimented with our price prediction system using the datasets mentioned in Section 2. The datasets were randomly partitioned into a training set (80%) and a test set (20%). We repeated such splits 3 times for each data set, and built separate models on each training set. (The low number of repetitions, 3, is caused by the computational requirements of each experiment.)

For our experiments, we used the decision tree implementation `rpart` [28] available in the statistical computing environment `R` [22] as a base learner. This implementation uses several parameters. The first parameter is a regu-

larization parameter *cp* that helps control the size of the trees. Any split that does not decrease the overall lack of fit by a factor of *cp* is not attempted by `rpart`. This parameter was set to 0.0005 for the Canon- and 0.0001 for the Nike dataset. The second parameter is the *maxdepth* parameter which was set to 2. Although trees of depth 2 are unable to model complex functions, an ensemble of such trees is a very flexible model.

We implemented the LS_Boost algorithm ourselves in `R` . The learning rate parameter $\nu$ was set to 0.1. The boosting algorithm was run until it became impossible to build an individual decision tree other than a single root node. Thus, we did not use a predetermined number of iterations $M$.

As a benchmark model, we used the most naive model possible: predicting the mean of the sales price in the training data per category.

To evaluate our models we use the error measures Mean Absolute Error (MAE) and Mean Relative Error (MRE):

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |\text{Predicted price}_i - \text{Observed price}_i|, \tag{2}$$

$$\text{MRE} = \frac{1}{N}\sum_{i=1}^{N} \frac{|\text{Predicted price}_i - \text{Observed price}_i|}{\text{Observed price}_i}. \tag{3}$$

[Table 2 about here.]

The experiments and the obtained results are summarized in Table 2. The errors reported in this table are *test set* errors. The reported errors clearly show that our system based on boosting outperforms the naive model of predicting the category mean price: for the Canon dataset the MAE is reduced from \$165 to \$72, for the Nike dataset it is reduced from \$22 to \$14. The reductions for the MRE are also substantial. However, we see that the MRE values reported are relatively high: 0.58 for Canon and 0.34 for Nike.

In the light of this observation, it is interesting to consider the cumulative distribution of the relative errors. These are shown in the top graphs of Figure 6. These graphs reveal that boosting predicts 57% of the Nike-auctions and 59% of the Canon auctions within a 20% range of the final auction price. For the naive method, these numbers are much lower: 33% and 24% respectively. When considering a 5% relative error range these numbers are 21% versus 7% for Canon and 21% versus 10% for Nike. So, although the average MRE values are high, a substantial number of auctions is predicted with reasonable accuracy.

[Figure 6 about here.]

As was explained in Section 3, relative importance plots visualize the importance of the indicators relative to one-another. Figure 7 shows these relative importance plots for the Canon- and Nike- auctions that we analyzed. The importances in these plots are averaged over the three experiments we performed.

The top part of Figure 7 shows the importance of the total item description (Dict.) versus the number of pictures (PICS) and the seller feedback rating (FB). It is clear that the item description is by far the most important predictor in both cases.

The bottom graphs in Figure 7 show the relative importances of the 52 most influential terms in the dictionary. The most important terms for the Canon data set were mostly technical terms as *ef* (extended focus), *CMOS* (a sensor which helps increases the quality of picture) and *powershot*. There are also important terms which identify models for example *EOS* and *XT* (Rebel XT series). The relative importance plot for the Nike data set shows a broad variety of split variables. Although a split term identifying one of

the existing Nike shoe models (*jordan*) is the most important, other terms like *deadstock* and *authentic*, are also influential.

[Figure 7 about here.]

## 5 Summary, Conclusions & Discussion

In this article we present a decision support system for predicting prices for online auctions. The predictions are based on a boosting model, which uses closed auctions of some product to predict prices for current auctions of the same product. The system uses the seller's feedback rating, the number of pictures on the web page and the seller's description of the item. The contribution of this study is twofold: it is the first study that uses the item descriptions in the prediction of eBay end prices, and it is the first study that uses boosting to this end. Boosting is based on combining decision trees, and therefore it is suitable for identifying important terms from a large term collection.

Gregg and Walczak have introduced an Auction Advisor system to support decisions for buyers and sellers [15]. Their support system summarizes several statistics about currently active and closed auctions. Our price prediction gives the user additional information. It would enable the Auction Advisor to leave out those items, for which the current bid exceeds the predicted end price.

We tested our price prediction model in a series of experiments. Interpretation yielded some interesting insights. Based upon the split variables, used the prediction model is able to identify influential terms in the description. These terms often relate to product subclasses and technical properties of the items and they are found without input of expert knowledge. In the

current system we are unable to identify the *directions* of these influences, but the system is easily extended towards this functionality.

In our experiments, the prediction model was capable of predicting approximately 21% of the auctions within a 5% range of the actual selling price. We remark that it may not be possible to achieve a much lower error. There are various reasons, such as bidding wars and sniping, why an item may be sold below or above its actual market value in practice. We can never model these effects fully, merely based on features that are available at auction start-time – part of the variance in the observed prices is 'intrinsic noise'. Nevertheless, it would be an attractive feature if we were to extend our system so that it would recognize when a prediction is likely to be accurate, and when it is likely to have a large error. This functionality could be added by using a bootstrap procedure, but we leave this for further research.

We believe that using a prediction model can prove profitable for a reseller on eBay. Another use of our system could be helping a seller by suggesting a reserve price or pointing him at terms in the item description he should use or avoid in order to raise revenue. Our system thus supports both buyers and sellers on eBay.

## References


[1] Yacine Atif. Building Trust in E-Commerce. *IEEE Internet Computing*, 6(1):18–24, Jan/Feb 2002.

[2] S. Ba. Establishing Online Trust through a Community Responsibility System. *Decision Support Systems*, 31(3):323–336, August 2001.

[3] S. Ba, A.B. Whinston, and H. Zhang. Building Trust in Online Auction Markets Through an Economic Incentive Mechanism. *Decision Support Systems*, 35(3):273–286, June 2003.

[4] Sulin Ba and Paul Pavlou. Evidence of the Effect of Trust Building Technology in Electronic Markets: Price Premiums and Buyer Behavior. *MIS quarterly*, 26(3):243–268, 2002.

[5] P. Bajari and A. Hortacsu. Economic Insights from Internet Auctions. *Journal of Economic Literature*, 42(2):457–486, 2004.

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

[7] L. Breiman. Arcing Classifiers. *Annals of Statistics*, 26(2):801–824, 1998.

[8] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees, 2nd Edition.* Chapman & Hall, NY, 1996.

[9] D. Bryan, D. Lucking-Reiley, N. Prasad, and D. Reeves. Pennies from eBay: The Determinants of Price in Online Auctions. Working Papers 0003, Department of Economics, Vanderbilt University, November 1999. May 2006 Version.

[10] eBay - Investor Relations. Web-site, 2006. `http://investor.ebay.com/`. Accessed on 6/20/2006.

[11] Y. Freund and R.E. Schapire. Experiments with a new Boosting Algorithm. In L. Saitta, editor, *Proceedings of the $13^{th}$ International Conference on Machine Learning*, pages 148–156, San Francisco, 1996. Morgan Kaufmann.

[12] Y. Freund and R.E. Schapire. A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[13] J.H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29(5):1189–1232, October 2001.

[14] R Ghani and H. Simmons. Predicting the End-Price of Online Auctions. Proceedings of *International Workshop on Data Mining and Adaptive Modelling Methods for Economics and Management* held in conjunction with the 15th European Conference on Machine Learning (ECML/PKDDD 2004), Pisa, Italy, 2004.

[15] D.G. Gregg and S. Walczak. Auction Advisor: an Agent-based Online Auction Decision Support System. *Decision Support Systems*, 41(2):449–471, 2006.

[16] Andrew J.I. Jones. On the Concept of Trust. *Decision Support Systems*, 33(3):225–232, July 2002.

[17] Vijay Krishna. *Auction Theory*. Academic Press/Elsevier Science, 2002. ISBN = 0-12-426297-X.

[18] A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes text Classification. In *Proceedings of AAAI-98, Workshop on Learning for Text Categorization*, 1998.

[19] Tetsuya Nasukawa and Tohru Nagano. Text Analysis and Knowledge Mining System. *IBM Systems Journal*, 40(4):967–984, 2001.

[20] Judith S. Olson and Gary M. Olson. i2i Trust in E-commerce. *Communications of the ACM*, 43:41–44, December 2000.

[21] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.

[22] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-07-0, `http://www.R-project.org`.

[23] P. Resnick, R. Zeckhauser, J. Swanson, and K. Lockwood. The Value Of Reputation On Ebay: A Controlled Experiment. *Experimental Economics*, 9(2):79–101, June 2006.

[24] D. Brian Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996.

[25] A.E. Roth and A. Ockenfels. Last-Minute Bidding and the Rules for Ending Second-Price auctions: Evidence from eBay and Amazon Auctions on the Internet. *American Economic Review*, 92(4):1093–1103, September 2002.

[26] G Salton. *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989. Isbn 0201122278.

[27] S. Slattery and M. Craven. Combining Statistical and Relational Methods for Learning in Hypertext Domains. In David Page, editor, *Proceedings of ILP-98, 8th International Conference on Inductive Logic Programming*, pages 38–52, Madison, US, 1998. Springer Verlag, Heidelberg, DE. Published in the "Lecture Notes in Computer Science" series, number 1446.

[28] T. M. Therneau, B. Atkinson, and B. Ripley. *rpart: Recursive Partitioning*, 2005. R package version 3.1-22, S-PLUS 6.x original at `http://www.mayo.edu/hsr/Sfunc.html`.

[29] Shanshan Wang, Wolfgang Jank, and Galit Shmueli. Explaining and Forecasting Online Auction Prices and their Dynamics using Functional Data Analysis. *Journal of Business and Economic Statistics*, 2006. Forthcoming.

[30] S.M. Weiss, C. Apté, F.J. Damerau, D.E. Johnson, F.J. Oles, T. Goetz, and T. Hampp. Maximizing Text-Mining Performance. *IEEE Intelligent Systems*, 14(4):63–69, 1999.

# List of Figures

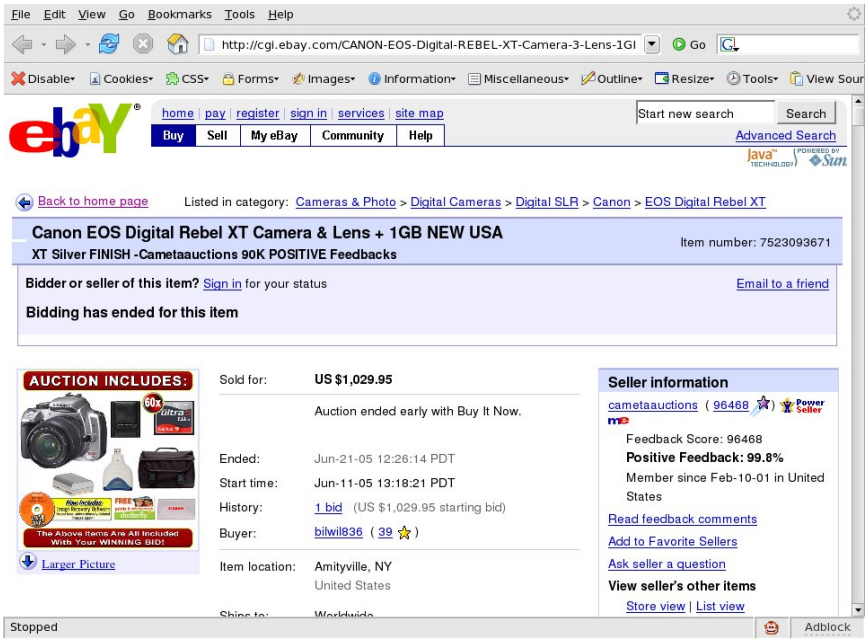


Figure 1: One of the downloaded eBay auction pages.

```
<html>

[non-relevant code omitted]

<td align=left valign=top nowrap>Winning bid: </td>
<td> <b> US $96.00 </b>

[non-relevant code omitted]

Positive Feedback: 99.5%

[non-relevant code omitted]

<!-- Begin Description -->
Great condition Canon Coolpix 2100 digital camera. Comes with an 8
megabyte compact flash card. Camera works perfectly with minor
blemishes.  The following are the features/specifications:
<xml:namespace prefix = o ns = urn:schemas-microsoft-com:office:office />
<o:p> </o:p>
</FONT> </SPAN> </P>
<!-- End Description -->

[non-relevant code omitted]

</html>
```

Figure 2: Relevant source code of an eBay auction page.

```
to diminish the size of the dictionari we us porter stem algorithm
thi algorithm strip the end from mani word in english on exampl of
thi strip i to convert plural to their singular form by strip the
letter s it i us a part of a term normal process that i usual done
when set up inform retriev system it assum that although these word
differ in quantiti thei do belong to the same term therefor we can
chang word to their root by strip off the suffix without chang what
i meant in figur you can read thi paragraph strip by porter stem
algorithm
```

Figure 3: A sample text stripped by Porter's stemming algorithm.

Terms

```
plural, normal, form, strip, suffix, retriev, belong, read,
singular, algorithm, figur, meant, letter, end, process, chang,
word, assum, usual, quantiti, system, stem, english, exampl,
term, root, set, convert, porter, inform, paragraph, diminish,
size, dictionari
```

Vector

```
(1,1,1,5,1,1,1,1,1,3,1,1,1,1,1,2,3,1,1,1,1,2,1,1,2,1,1,1,2,1,1,1,1,1)
```

Figure 4: Vector representation of the sample text.

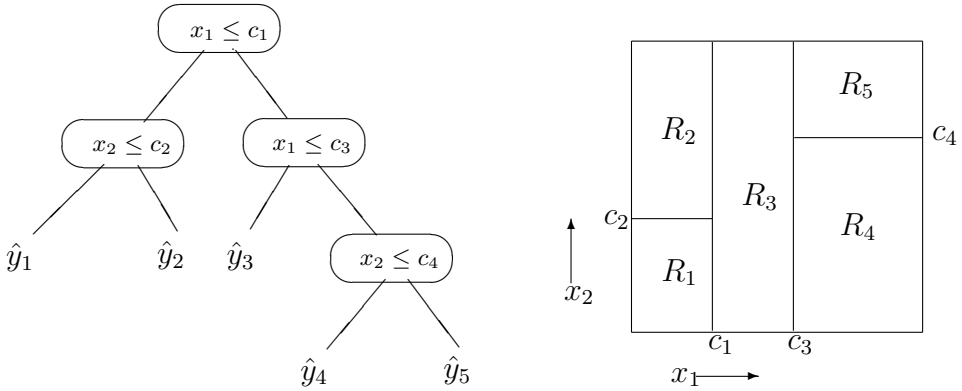


Figure 5: A decision tree for a dataset with two explanatory variables(left), and the corresponding partitioning of the feature space (right). For each leaf $\ell$ and each corresponding region $R_\ell$ the estimate of the target value is the average $\hat{y}_\ell$ of the observed $y$ values within that region.

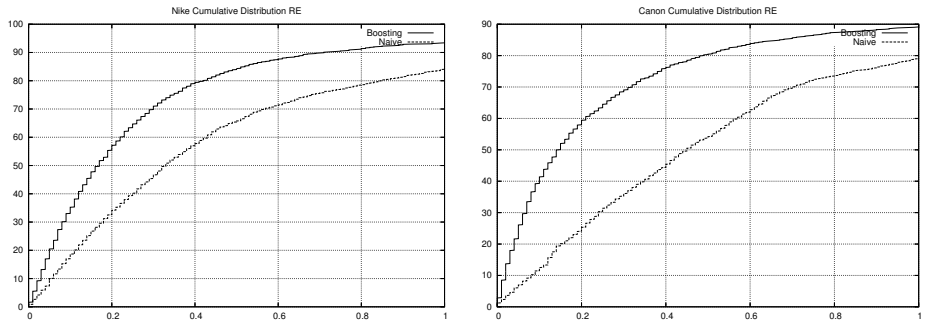


Figure 6: Distribution of the errors in the datasets. The horizontal axis values for the relative error, the vertical axis shows the percentage of auctions in the test set predicted with a relative error below that value.

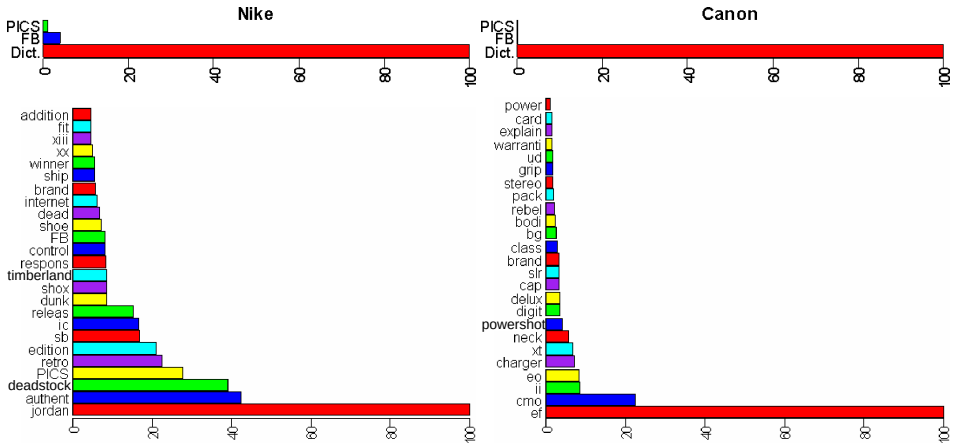


Figure 7: Relative importance plots for Nike (l) and Canon (r) datasets. The upper two graphs show the importance of the total item description (Dict.) versus the number of pictures (PICS) and the seller feedback rating (FB). In the lower graphs we displayed the relative importance of the most important terms in the dictionary.

## List of Tables

| | **Nike** | **Canon** |
|---|---|---|
| Download period | August 2005 | August 2005 |
| Downloaded auctions | 5945 | 5042 |
| Position upper whisker (price) | 163.5 | 1175 |
| Position lower whisker (price) | 5 | 5 |
| Non-outlier auctions | 5546 | 4603 |
| Average selling price ($) | 60.61 | 355.86 |
| Lower bound word occurrence ($f_l$) | 50 | 80 |
| Number of words in dictionary | 1258 | 1926 |

Table 1: Summary of the data sets on auctions of Nike men shoes and Canon cameras. The whiskers refer to the box-plot used for filtering outliers. The parameter $f_l$ was set in a series of preliminary experiments.

| | **Nike** | **Canon** |
|---|---|---|
| data set size | 5546 | 4603 |
| training set size | 4437 | 3683 |
| test set size | 1109 | 920 |
| number of repetitions | 3 | 3 |
| *cp* | 0.0001 | 0.0005 |
| *maxdepth* | 2 | 2 |
| avg number of boosting iterations | 2538 | 609 |
| $\nu$ | 0.1 | 0.1 |
| avg/min/max MAE Boosting | 14.18/13.69/14.90 | 71.99/69.62/74.46 |
| avg/min/max MRE Boosting | 0.343/0.32/0.38 | 0.585/0.58/0.59 |
| avg/min/max MAE Naive | 22.1/21.74/22.71 | 165.69/163.22/170.02 |
| avg/min/max MRE Naive | 0.55/0.52/0.58 | 1.2/1.18/1.25 |

Table 2: Summary of the performed experiments and the obtained results. The reported errors are average, minimal and maximal error over the three repetitions. All errors are *test set errors*.