Predicting The Best Starting Price for Ebay Auctions

[Put the type of article here using command `\articletype{type of article}`**.]**

Jiacheng Liao[1], Yi Wan[1], Shuang Zhou[1], Zhaoyin Zhu[2]

1 Department of Computer Science, New York University, New York, NY 10012, USA

2 Division of Biostatistics, NYU School of Medicine, New York, NY 10016, USA

Abstract: Online auctions are one of the most popular methods to buy and sell items on the internet. With more than 100 million active users globally, eBay is the worlds largest online marketplace, where anyone can buy and sell anything. In order to successfully selling products on Ebay, a reasonable starting price does not only determine whether the product will be sold or not but also affects the profit you can make from the transaction. In this project, we use the historical auction data collected from eBay from April 2013 to the first week of May 2013 which contains information about 296,048 successful and unsuccessful auctions. Different statistical models and machine learning algorithms will be utilized to study online auction patterns and predict the starting price that maximizes profits. Furthermore, we will compare the performance of different methods and summarize the pros and cons in different situations.

Keywords: Ebay Auction • Predictive Analytics • Data Mining

1. Introduction (Data and Business Understanding)

EBay is the worlds largest marketplace for sports autographs, the vast majority of the sites membership uses it to buy and/or sell items via auction format. The ability to provide a method to estimate auction sale prices is desirable to this community. Members of most communities related to collectibles have reported they most often try to predict how much an auction would sell for by performing a search for item and manually calculating the average sales price. In this project, our first objective is to determine whether an auction listing will result in a sale. In addtion, we aim to predict the final sales price as well as the best starting price using data mining techniques.

2. Data Preparation

Data Preparation is an crucial and time-consuming part of our data mining project. It involves selecting data to include, cleaning data to improve data quality, constructing new data that may be required, integrating multiple data sets, and formatting data. We first preprocessed the downloaded data sets using shell scripts. To gain more experience on feature selection, we have performed the selection procedure on all three tools used in the class, RapidMiner, Weka and R. With different algorithms and parameters, we get slightly different but generally consistent results.

2.1. Data Preprocessing

Before any data can be used for later feature reduction and selection, we preprocessed our data sets. Here the tool we use is shell scripts. Initially, the raw data consists of training sets and test sets. Since we intend to do cross-validation on the whole data set, we merge all the separate data sets as one complete sets. Then we carefully go through all the data attributes and deleted all that obviously contain meaningless or irrelevant information to our analysis, such as ebayID, sellerName, etc. In addition, some attributes with ambiguous meanings are also discarded.

2.2. Feature Selection Using Weka

First, we perform feature selection in Weka. To enable Weka to better handle the data, the first step is establishing a nominal class label. Weka comes with an unsupervised discretization method that allows users to discretize attributes, and after applying it to the class field in our dataset, the originally numeric type with value 0 and 1 becomes nominal labels “0” and “1”.

For feature selection, two algorithms are used in Weka: Information Gain and Feature Subset Selection. We first used Information Gain Algorithm and the result is as below. According to the information gain results we have, if we set the entropy threshold to 0.95, we will need the top 8 features.

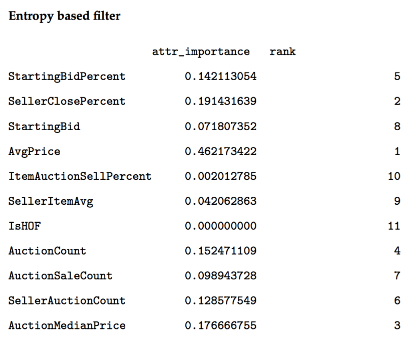
```
Ranked attributes:
 0.318735   3 SellerClosePercent
 0.25068    2 StartingBidPercent
 0.20478   11 SellerAuctionCount
 0.119311   4 StartingBid
 0.07023    7 SellerItemAvg
 0.041762  12 AuctionMedianPrice
 0.040966   5 AvgPrice
 0.031772   9 AuctionCount
 0.028263  10 AuctionSaleCount
 0.005359   6 ItemAuctionSellPercent
 0.000207   8 IsHOF_1
```

Using the number of features obtained above, we set the number of features to 8 and try feature subset selection. We start with empty subset, and we use CfsSubsetEval as the evaluation of each subset, and we use GreedyStepWise search method, which basically select the best next feature based on current subset. Here is the result we get.

```
Ranked attributes:
 0.1038   3 SellerClosePercent
 0.1328   2 StartingBidPercent
 0.1245   4 StartingBid
 0.1138   9 AuctionCount
 0.1076  11 SellerAuctionCount
 0.1023  12 AuctionMedianPrice
 0.0983   6 ItemAuctionSellPercent
 0.0931   8 IsHOF_1
```

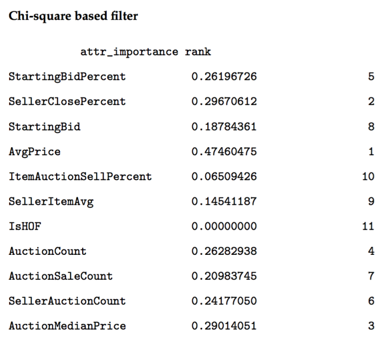
2.3. Feature Selection Using R

The caret R package provides tools automatically report on the relevance and importance of attributes in your data and even select the most important features for you. Here we perform three different feature selection method on our dataset, namely entropy based filter, Chi-square based filter and Correlation based filter. We apply all three filters to get a better sense of what attributes are more important, and we have the results below.



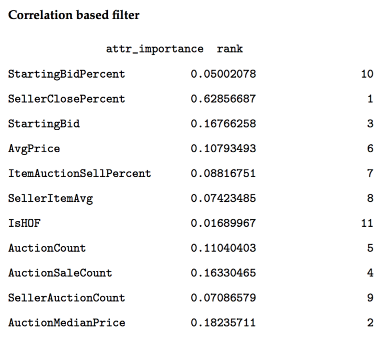
Entropy based filter

```
                        attr_importance   rank
StartingBidPercent          0.142113054       5
SellerClosePercent          0.191431639       2
StartingBid                 0.071807352       8
AvgPrice                    0.462173422       1
ItemAuctionSellPercent      0.002012785      10
SellerItemAvg               0.042062863       9
IsHOF                       0.000000000      11
AuctionCount                0.152471109       4
AuctionSaleCount            0.098943728       7
SellerAuctionCount          0.128577549       6
AuctionMedianPrice          0.176666755       3
```



Chi-square based filter

```
                        attr_importance rank
StartingBidPercent          0.26196726     5
SellerClosePercent          0.29670612     2
StartingBid                 0.18784361     8
AvgPrice                    0.47460475     1
ItemAuctionSellPercent      0.06509426    10
SellerItemAvg               0.14541187     9
IsHOF                       0.00000000    11
AuctionCount                0.26282938     4
AuctionSaleCount            0.20983745     7
SellerAuctionCount          0.24177050     6
AuctionMedianPrice          0.29014051     3
```



Correlation based filter

```
                        attr_importance   rank
StartingBidPercent          0.05002078      10
SellerClosePercent          0.62856687       1
StartingBid                 0.16766258       3
AvgPrice                    0.10793493       6
ItemAuctionSellPercent      0.08816751       7
SellerItemAvg               0.07423485       8
IsHOF                       0.01689967      11
AuctionCount                0.11040403       5
AuctionSaleCount            0.16330465       4
SellerAuctionCount          0.07086579       9
AuctionMedianPrice          0.18235711       2
```

As a result, we summarize all the three different methods and compute average rank for each attribute. The result is presented in table 1.

Table 1. **Feature Selection Using R Result**

Attribute Name	Entropy based filter	Chi-square based filter	Correlation based filter	Average Rank
SellerClosePercent	2	2	1	1.7
AvgPrice	1	1	6	2.7
AuctionMedianPrice	3	3	2	2.7
AuctionCount	4	4	5	4.3
AuctionSaleCount	7	7	4	6.0
StartingBid	8	8	3	6.3
StartingBidPercent	5	5	10	6.7
SellerAuctionCount	6	6	9	7.0
SellerItemAvg	9	9	8	8.7
ItemAuctionSellPercent	10	10	7	9.0
IsHOF	11	11	11	11.0

2.4. Feature Selection Using RapidMiner

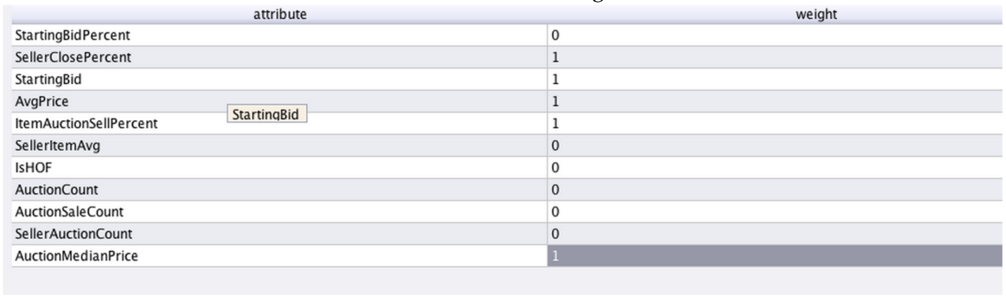
First we choose a feature selection method from RapidMiner. Here we use Forward Selection. Forward selection operator selects the most relevant attributes of the given ExampleSet through a highly efficient implementation of the forward selection scheme.

Forward selections uses wrapper method to select attributes. Basically, it starts with an empty attribute set. It them add one attribute to run the model and measures the performance. The process keep adding attributes to the model to see if there is performance gain. Depending on the parameter set, it will terminate until there is no performance improvement or no significant performance improvement. Here we also use cross-validation to measure the performance of the model as well as the current attributes set the operator.

For predicting the whether the given item can be sold or not, we use several classification algorithms, including: Naive Bayes, Decision Tree, Random Forest, Rule Induction, Neural net, Logistic Regression, Support Vector Machine. We will consider efficiency and prediction accuracy for each model to decide which one to adopt.

2.4.1. Naive Bayes modeling

A. The attribute weight



attribute	weight
StartingBidPercent	0
SellerClosePercent	1
StartingBid	1
AvgPrice	1
ItemAuctionSellPercent	1
SellerItemAvg	0
IsHOF	0
AuctionCount	0
AuctionSaleCount	0
SellerAuctionCount	0
AuctionMedianPrice	1

B. The prediction model result and auc curve

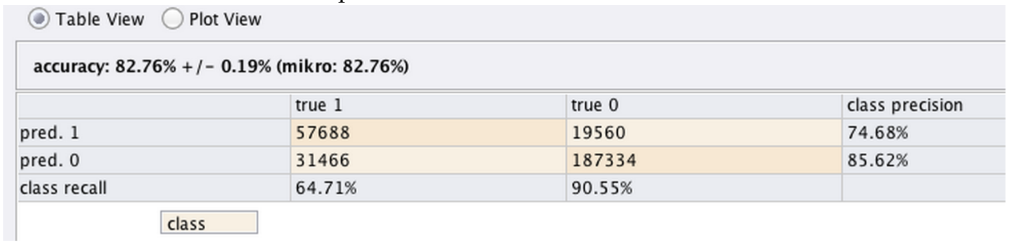
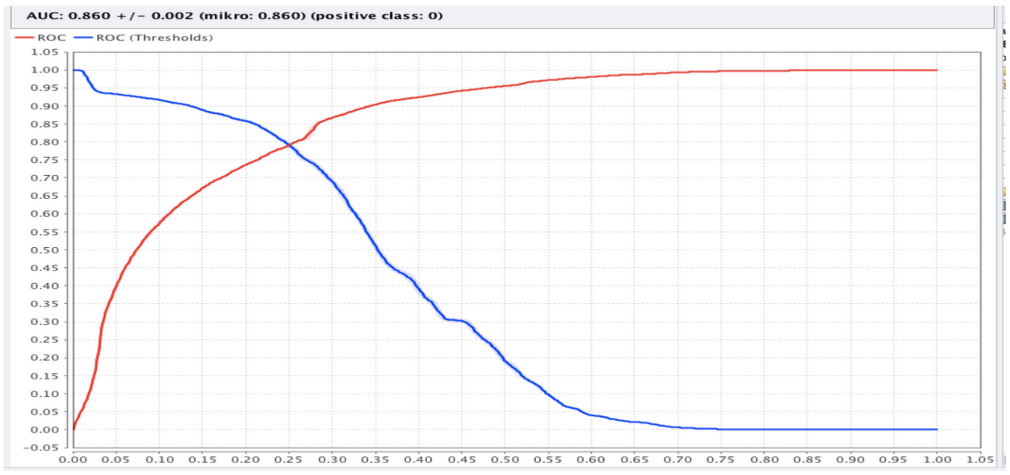


Table View
Plot View

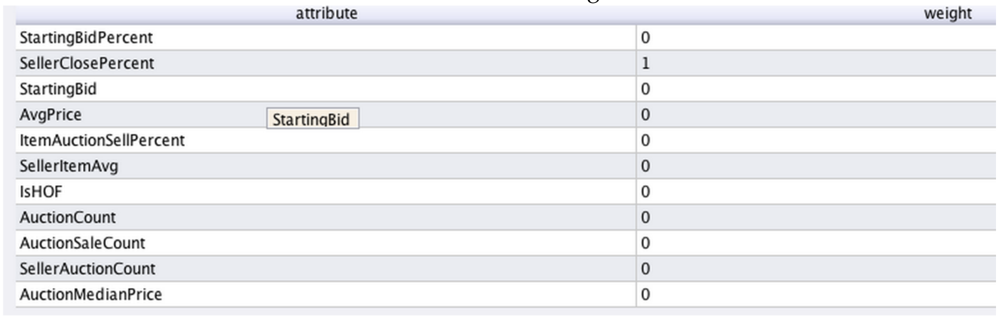
accuracy: 82.76% +/- 0.19% (mikro: 82.76%)

	true 1	true 0	class precision
pred. 1	57688	19560	74.68%
pred. 0	31466	187334	85.62%
class recall	64.71%	90.55%	



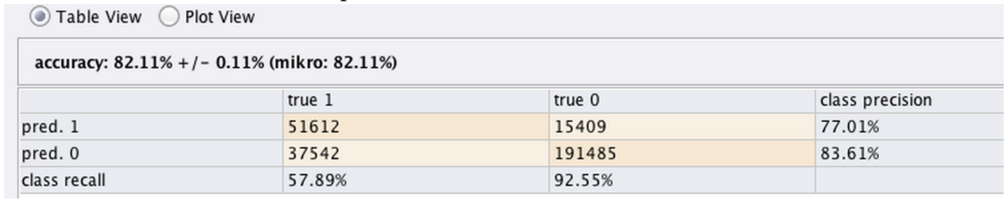
2.4.2. Decision Tree modeling

A. The attribute weight


StartingBid

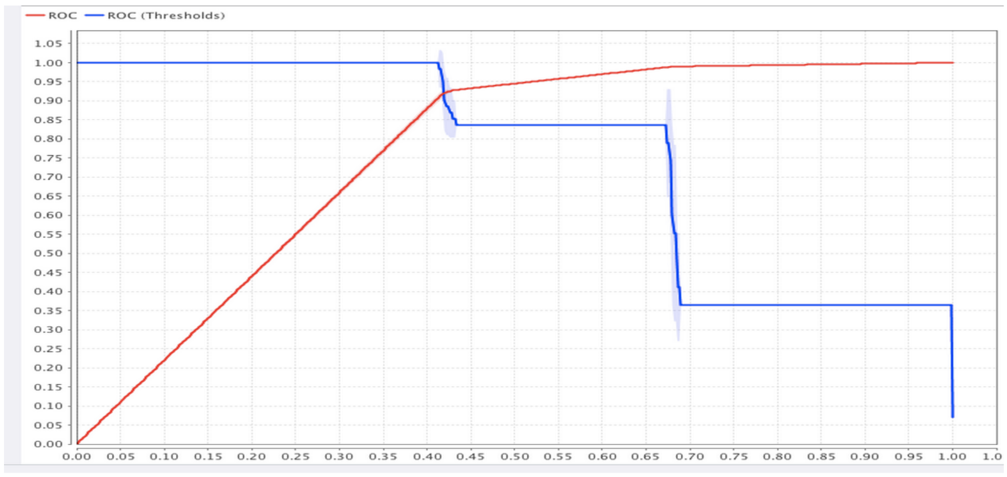
attribute	weight
StartingBidPercent	0
SellerClosePercent	1
StartingBid	0
AvgPrice	0
ItemAuctionSellPercent	0
SellerItemAvg	0
IsHOF	0
AuctionCount	0
AuctionSaleCount	0
SellerAuctionCount	0
AuctionMedianPrice	0

B. The prediction model result and auc curve



accuracy: 82.11% +/- 0.11% (mikro: 82.11%)

	true 1	true 0	class precision
pred. 1	51612	15409	77.01%
pred. 0	37542	191485	83.61%
class recall	57.89%	92.55%	



2.5. Other Trials

The other methods including Random Forest, Rule Induction, Neural net, Logistic Regression, Support Vector Machine takes a very long time to finish (more than one hour). Since the dataset we are using is not very large, we consider them not suitable for our problem.

2.6. Final Feature Selection

To fairly select the features based on all three tools, we define an algorithm to integrate the results. An attributing score is assigned to each feature for final ranking, and the rules for assigning the score is as follow:

1. In ranked result, the feature ranked ith is assigned a score of $12 - i$, e.g. the highest feature gets a score of 11.
2. In not ranked result, the features selected as useful is assigned a score of 8 and others are assigned score of 0.
3. The final score is a summation of the results from all tools with all algorithms.

The result computed is presented in table 2.

Table 2. Overall Feature Attributing Scores

Attribute Name	Weka Attributing Score	R Attributing Score	RapidMiner Attributing Score	**Overall Attributing Score**
SellerClosePercent	18	22	16	56
AuctionMedianPrice	12	20	8	40
StartingBid	18	12	8	38
AvgPrice	7	20	8	35
StartingBidPercent	21	10	0	31
AuctionCount	13	16	0	29
SellerAuctionCount	17	8	0	25
AuctionSaleCount	6	14	0	20
ItemAuctionSellPercent	7	4	8	19
SellerItemAvg	8	6	0	14
IsHOF	5	2	0	7

As indicated by Weka, the number of features we need to achieve a 0.95 information gain is 8, we narrowed the features down to the first 8 features in the table. And from here on, unless specified otherwise, we are using the reduced dataset with only these features.

3. Modeling

Modeling involves selecting suitable modeling techniques, generating test designs to validate the model, building predictive models and assessing these models.

A predictive model is a mathematical function that predicts the value of some output variables based on the mapping between input variables. Historical data is used to train the model to arrive at the most suitable modeling technique. For example, a predictive model might predict the risk of developing a certain disease based on patient details. Some commonly used modeling techniques are as follows: Regression analysis that analyzes the relationship between the response or dependent variable and a set of independent or predictor variables. Decision trees that help explore possible outcomes for various options. Cluster analysis that groups objects into clusters to look for patterns. Association techniques that discover relationships between variables in large databases.

Modeling in this project consists of two parts. In the first part, we try to build a model to predict whether an item can be sold or not. And in the second model, we try to predict a best starting price that maximize the final sale price.

3.1. Modeling for Sale / No Sale

In all three tools here, we tried to use different type of classification algorithms to decide whether items can be sold or not based on some information of item across Ebay.

3.1.1. Modeling in Weka

To compare the effectiveness of different algorithms, we decided to use F measure as the criteria to measure performances of each model. The label of interest is sold (denoted by 1) and not sold (denoted by 0). NB classifier is the first algorithm we try. Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. It's as straightforward as it gets.

Another algorithm we use is Logistic Regression, Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. It fits very well in our situation, and it's a fast model to train and use. Both NB classifier and Logistic Regression are applied over the dataset in Weka, and results are presented in table 3.

Table 3. **Model Comparison in Weka**

Model Name	F measure
Naive Bayes	0.639
Logistic Regression	0.674

Obviously Logistic Regression model works better than NB model for this dataset, with a F measure of 0.674. But we will continue to examine different models with other tools.

3.1.2. Modeling in RapidMiner

In RapidMiner, models we tried include Naive Bayes, Random Forest, Decision Tree, Logistic Regression, Neural Network. Some of the models can't finish in reasonable time due to the RapidMiner design that enforce us to add "Performance" module into the process, which noticeably increases the running time. So as a result, we only present Naive Bayes, Random Forest, and Decision Tree models here.

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. The decision tree algorithm comes with a "depth" parameter and a evaluation algorithm. For the evaluation algorithm, we used Gain Ratio.

Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set. See table 4 for all results.

Table 4. **Model Comparison in Weka**

Model Name	F measure
Naive Bayes	0.629
Decision Tree (Depth = 5)	0.6223
Decision Tree (Depth = 10)	0.6334
Decision Tree (Depth = 50)	0.6334
Random Forest	0.6112

In Decision Tree classifier, to obtain the best result, we tried multiple depths from 5 to 50, the accuracy grows a little with the depth at first, and then stays the same when the depth is greater than 10. Intuitively this makes sense because as the depth go up, it's more likely to find relations between each attribute and the label, but given there're only 8 features in total, the depth wouldn't matter too much after it exceeds the number of features.

In Random Forest algorithm, a somewhat surprising discovery is that the overall accuracy is slightly lower than Decision Tree algorithm, because given that Random Forest classifier is a much more complicated algorithm, we suppose it should do better. But as a property of Random Forest, it tends to get rid of overfitting behaviors of decision trees, and the accuracy here is computed based on the exact same dataset we used for training the classifier. Therefore, this is an acceptable loss. And we will find out which one is more accurate when we apply them on unknown datasets.

3.1.3. Modeling in R

Just to gain experience on as many tools as possible, we also try to build our model with R. We chose two algorithms to experiment with, namely, Logistic Regression (just to compare results with RapidMiner and Weka) and K-Nearest Neighbors. The k-Nearest Neighbors algorithm (or K-NN for short) is a non-parametric method used for classification and regression. In most cases, the input consists of the k closest training examples in the feature space. The output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor. We are able to find Logistic Regression algorithm from the famous "Generalized Linear Model" package in R, and we also find a very authoritative implementation of KNN algorithm in ________. Logistic Regression Result from R is in table 5:

Table 5. **Coefficient Table for Logistic Regression**

Entry Name	Coefficient
Intercept	-1.646e+00
StartingBidPercent	-1.120e+00
SellerClosePercent	4.857e+00
AvgPrice	-4.210e-04
AuctionCount	-1.623e-03
AuctionSaleCount	5.103e-03
SellerAuctionCount	2.008e-05
AuctionMedianPrice	3.194e-03

To interpret the table, let's say $e_1, e_2, ..., e_n$ are n entries (features), and $c_0, c_1, c_2, ..., c_n$ are n corresponding coefficients with c_0 being the intercept, or constant, in this formula, computed by the algorithm. Then, for a given instance x, the label of this instance can be generated with:

$$L(x) = c_0 + \sum_{i=1}^{n} e_i \times c_i$$

We can't obtain much information by simply looking at the expression, although some may wonder the importance of each feature is indicated by the absolute value of each coefficient. That's not true in most cases, as all features have different magnitudes and coefficients play the role as mitigator among all those features. We evaluate the model, and the F measure reaches 0.674.

For K-NN, we try to run it with different number of neighbors in order to find the "magic number", and results are presented in table 6.

Table 6. **K-NN With Different Number Of Neighbors**

Number of Neighbors	F measure
5	0.639
6	0.635
7	0.636
8	0.632
9	0.633
10	0.630
15	0.627
20	0.627
50	0.621

It appears that when the number of neighbors is from 5 to 10, the accuracy achieves the highest, and as the number keeps growing, the performance actually goes down a little, although the loss is very minimal. Therefore we try all values from 5 to 10 and we find that "7" and "9" out-perform other values. Considering the fact that the smaller the number of neighbors is, the (slightly) faster will the algorithm runs, we will choose "7" as the parameter for

K-NN model in this project.

3.1.4. Final Model Selection

Considering all models we've tried, we decide to go with Logistic Regression model. It performs the most stably and achieves the highest percent accuracy according to our cross-validation result from R.

3.2. Modeling For Best Starting Price

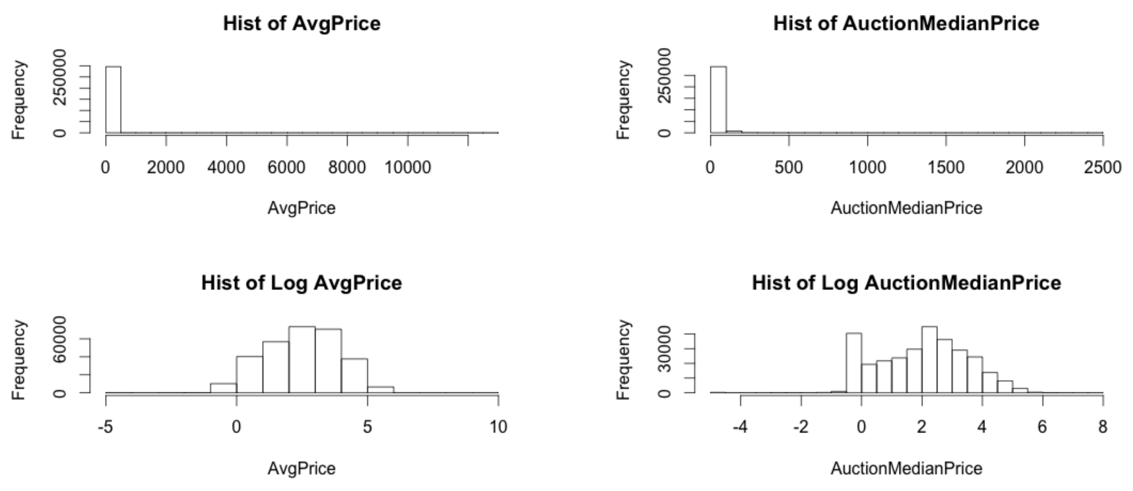
For better flexibility and simplicity of debugging, we decide to adopt R for the whole process. For the choice of model, Linear Regression fits almost perfectly into our requirement. Motivations that drive us to this model includes:

- As the final price is a continuous numerous value, any classification algorithm that deals with discrete values require extra work of partitioning.
- Linear Regression provides a very user-friendly representation of the model that can leverage some insights of the data.
- The final price field has a very wide range of values, it will be very hard to distribute instances evenly and maintain similar width of each class at the same time.

One particularly interesting thing we do before applying the model to our dataset is to normalize the data. For any continuous covariants in the input to Logistic Regression or Linear Regression, the models expect the a Normal Distribution of it, thus we apply a log transformation on the dataset. The log transformation can be used to make highly skewed distributions less skewed. This can be valuable both for making patterns in the data more interpretable and for helping to meet the assumptions of inferential statistics. After investigation of "skewed distributed" fields, we decide to apply a log transformation on Average Price and Median Price. The transformations are illustrated below.

AveragePrice and LoggedAveragePrice

MedianPrice and LoggedMedianPrice



After transformation, both originally very skewed fields becomes approximately normal distributed. Therefore, now we can apply Linear Regression on our dataset. Results are shown in 7: [1]

Table 7. **Coefficient Table for Linear Regression**

Entry Name	Coefficient
Intercept	-1.165e-01
StartingBidPercent	1.000e+00
SellerClosePercent	6.523e-01
LoggedAvgPrice	-2.689e-03
AuctionCount	-1.469e-04
AuctionSaleCount	5.276e-04
SellerAuctionCount	9.239e-06
LoggedAuctionMedianPrice	2.041e-02

And as always, we use cross-validation for testing the performance. To show the effectiveness of the log transformation, we run two evaluations on the model before and the model after. Results can be found in 8.

Table 8. **Performance for Linear Regression**

	Model Without Log	Model With Log
Bias	0.161	0.161
Standard Deviation	0.0015	0. 0017
Mean Square Error	0.139	0.138

Based solely on these results, the model with log transformation only performs slightly better than the model we have before, but we find it particularly useful when applied on test data. We will describe it in the next section.

[1] *Actually, we adopted log transformation in our final version of Logistic Regression as well, and we will use it in our deployment.*

4. Evaluation

We've built our model based on a dataset containing auctions ending in April 2013, and in this section, we will use the test dataset which contains 7460 auctions ending in first week of May 2013 to evaluate the models we obtained from previous sections.

For logistic model, we use precision, recall and F-measurement to assess the performance. Precision (also called positive predictive value) is the fraction of retrieved instances that are relevant, recall (also known as sensitivity) is the fraction of relevant instances that are retrieved and F measure f is defined as

$$f = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The results with test dataset is shown in table 9.

Table 9. Evaluation of logistic model with test dataset

	Without Log	With Log
Precision	0.848	0.858
Recall	0.526	0.528
F-measurement	0.643	0.654

For simple linear model, we used absolute bias to represent the accuracy and standard deviation to represent the precision of prediction on test dataset. Mean square error (MSE) = Bias^2 + Variance was reported as well to balance bias and variance. The results with test dataset is shown in table 10.

Table 10. Evaluation of simple linear model with test dataset

	Without Log	With Log
Bias	0.151	0.152
Standard Deviation	0.0017	0.0016
MSE	0.330	0.266

Here we can observe a big performance boost when we use log transformation, with a 21% MSE rate drop.

5. Deployment

After we land on our models, we decide to deploy a Java backed WebPage to provide an interactive service. The system allows users to query about their items for prediction. Users provide necessary information and AngularJS and Bootstrap are used for web design, and requests are passed to the server for realtime classification. The backend starts with a model trained with training data, and simply classifies every instance coming in.

For algorithm implementations, we use Weka JAR files, which can do basically everything Weka itself can do. The server hosts on CIMS machines and can be monitored remotely.

6. Conclusion (ToDo)

TODO

Acknowledgements

The author(s) would like to thank some institutions for support and so on.

References

[1] Han, Jiawei, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques: concepts and techniques*. Elsevier, 2011.

[2] Rajaraman, Anand, and Jeffrey D. Ullman. *Mining of massive datasets*. Vol. 77. Cambridge: Cambridge University Press, 2012.

[3] Bari, Anasse, Mohamed Chaouchi, and Tommy Jung. *Predictive analytics for dummies*. John Wiley & Sons, 2014.

[4] Bari, Anasse. Predictive Analytics Course Lecture Notes. 2015 Fall.

[5] Brownlee, Jason. Feature Selection with the Caret R Package. http://machinelearningmastery.com/feature-selection-with-the-caret-r-package/

[6] Grossman, Jay. Predicting eBay Auction Sales with Machine Learning.
Retrived from http://jaygrossman.com/post/2013/06/10/Predicting-eBay-Auction-Sales-with-Machine-Learning.aspx